Ultra Wide Band Multiple Access Performance Using TH-PPM and DS-BPSK Modulations

THESIS

Courtney M. Canadeo, Second Lieutenant, USAF

AFIT/GE/ENG/03-06

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense or U.S. Government.

AFIT/GE/ENG/03-06

Ultra Wide Band Multiple Access Performance Using TH-PPM and DS-BPSK Modulations

THESIS

Presented to the Faculty of the Graduate School of Engineering and Management
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

Courtney M. Canadeo, B.S.E.E.
Second Lieutenant, USAF

March, 2003

AFIT/GE/ENG/03-06

Ultra Wide Band Multiple Access Performance Using TH-PPM and DS-BPSK Modulations

Courtney M. Canadeo, B.S.E.E.

Second Lieutenant, USAF

Approved:

Dr. Michael A. Temple Thesis Advisor	6 Mar Ø3 Date
Rusty O. Bald Dr. Rusty O. Baldwin Committee Member	Ø6 Mar Ø3 Date
Richd A Raines Dr. Richard A. Raines Committee Member	6 Mar 03 Date

Acknowledgements

“You got to be careful if you don’t know where you’re going, because you might not get there.” —Lawrence ‘Yogi’ Berra

Thank you, Dr. Temple, for the invaluable guidance you provide.

Courtney M. Canadeo

Table of Contents

List of Figures

Figure Page

Figure Page

List of Tables

AFIT/GE/ENG/03-06

Abstract

The increasing demand for portable, high data rate communications has focused much attention on wireless technology. Ultra Wide Band (UWB) waveforms have the ability to deliver megabits of information while maintaining low average power consumption. In accordance with recent FCC rulings, UWB systems are now allowed to operate in the unlicensed spectrum of 3.1 to 10.6 GHz, motivating renewed interest in the forty year old concept of impulse radio.

Gaussian monocycles produce UWB waveforms occupying large bandwidths with multiple access (MA) capability enabled by spread spectrum techniques. Time Hopping (TH) and Direct Sequence (DS) modulations are considered here for UWB MA applications. This work extends Gold coding results and characterizes UWB performance using Simulated Annealing (SA) and Random Integer (RI) codes for TH and DS UWB applications. TH-PPM and DS-BPSK performance is evaluated using simulated probability of bit error P_b under MA interference (MAI), multipath interference (MPI), and narrow band interference (NBI) conditions for synchronous and asynchronous networks.

Communication performance is validated for a single user operating over an AWGN channel and extended to incorporate MA capability. For a synchronous network of 15 users, Gold coded TH-PPM yields average MA BER improvement factors of -17.1 dB and -5.64 dB relative to RI and SA codes. Gold coded DS-BPSK provides an improvement factor of -18.9 dB (RI) and -26.3 dB (SA). For an asynchronous network, Gold coded TH-PPM yields an improvement of -4.07 dB over RI and a loss of 1.50 dB for SA, while Gold coded DS-BPSK yields a loss of 0.30 dB for RI and 0.48 dB for SA. For a single transmitter/receiver link, P_b increases by a factor of 1.5 (TH-PPM) and 25 (DS-BPSK) per multipath replication.

Ultra Wide Band Multiple Access Performance Using TH-PPM and DS-BPSK Modulations

I. *Introduction*

1.1 Motivation for Using Ultra Wide Band

1.1.1 Wireless Trade-Offs. The world continues to increase its dependence on electronic communications. Balancing desired properties for data transmission requires development of new methods for exchanging information. Ideally, large quantities of data are rapidly transmitted by many users, simultaneously, over a significant distance. Unfortunately, these characteristics are in competition with each other and trade-offs must occur to obtain the best solution for a particular application.

Wireless communications have become popular because they address growing demands. Portable wireless devices permit high data rates at low cost and, with improved semiconductor technology, low power consumption. Crowding within existing spectral allocations is driving the need for new ways to efficiently use available frequency bands. The increase in high-speed, wired access to the Internet has increased the demand for high-speed communications within the home. The need for robust forms of transmission that do not interfere with other users, even inside relatively small areas, such as a single room in a building, is a pressing requirement. Ultra wide band (UWB) technology is a form of wireless communications and is becoming a popular choice for addressing these types of issues.

1.1.2 Applications. The field of ultra wide band signalling is just beginning to appear across a vast number of applications where the technology can improve existing systems or provide entirely new capabilities. The following sections present ideas that have been proposed using UWB or, in some cases, ideas that have already been implemented.

1.1.2.1 Advanced Radar Sensing. Ultra wide band signalling can be implemented with a large bandwidth at relatively low frequencies, making it suitable for through-wall imaging and radar sensing applications. Lower frequency components enhance signal propagation through the ground or walls while the larger bandwidth provides higher resolution. This further limits the post-processing needed to intelligently view a radar image and reduces cost and complexity. Many UWB systems can now resolve multipath interference to within one-half of the pulse length. Thus, for a 500 picosecond pulse travelling at the speed-of-light in free-space, an image could be produced with approximately 8 centimeter resolution. In addition, recent studies [14] have indicated that UWB may be more prone to edge diffraction and propagation through large cracks. Such sensing could also be used to locate people within rubble. Hospitals are considering using portable UWB technology so doctors can monitor patient information in a dynamic, remote environment.

1.1.2.2 Precision Location. The use of GPS provides location data within meters. For outdoor tracking and large scale identification this accuracy is acceptable. GPS satellite signals work well in outdoor applications. However, both signal strength and location resolution are severely degraded indoors. The centimeter level precision and multipath mitigation available with UWB makes it more suitable for the indoor environment. Rescue services could benefit by using UWB technology to locate people inside buildings during emergencies. On a more routine basis, UWB nodes could be installed throughout a building to electronically track people carrying UWB sensors - providing transfers of information throughout a building or adjusting environmental factors to personal preference.

1.1.2.3 Inventory Control. During Desert Storm in 1991, the U.S. Navy shipped 60% of its containers with inaccurate or missing paper manifests accounting for a $3 billion loss [14]. Ideas have been proposed to use small UWB transmitters attached to inventory items to periodically transmit a low power signal to a central database. This information could be used to maintain an accurate inventory and establish a map to locate items and aid in determining the most efficient arrangement to maximize productivity. With the centimeter level precision afforded by UWB, robots could automate the task of inventory retrieval or movement.

1.1.2.4 "Smart" Homes. Several countries have been investigating new ways to increase platform portability of wireless communications. Japan, for instance, is using IEEE 1394 digital cabling to transmit audio and visual signals in and out of homes. Within the home, the idea is to connect all multimedia equipment via a UWB design, eliminating the need for wires throughout the building. Of course, security remains a concern and research is on-going to develop secure encryption techniques.

1.1.2.5 Collision Avoidance. The FCC has established emissions level requirements for vehicular radar systems. One UWB idea equips vehicle bumpers with UWB sensors. When another vehicle approaches too closely, the sensor would take over vehicle braking and/or steering as needed to prevent a collision.

1.1.2.6 Internet Mobility. The use of wireless communications for Internet applications is gaining popularity. Current technologies such as IEEE 802.11, Bluetooth, and the European HiperLAN, allow mobile users to access the Internet at high data rates. Efforts are underway to find methods to allow the simultaneous operation of these technologies.

1.2 Problem Statement and Scope

In accordance with recent FCC rulings, UWB systems are now allowed to operate in the unlicensed spectrum of 3.1 to 10.6 GHz, motivating renewed interest in the forty year old concept of impulse radio. By design, the narrow pulse of Gaussian monocycles yields UWB waveforms occupying relatively large bandwidths. To enable multiple access (MA) capability, spread spectrum communication techniques are employed. Time Hopping (TH) and Direct Sequence (DS) modulations are two methods commonly considered for UWB MA applications. Questions remain regarding the validity of typical assumptions used for modeling auto- and cross-correlation characteristics of spreading codes [11],[32]. This work extends Gold coding results and characterizes UWB performance using Simulated Annealing (SA) and Random Integer (RI) codes for TH and DS UWB applications. It also evaluates Time Hopping Pulse Position Modulation (TH-PPM) and Direct Sequence Binary Phase Shift Keying (DS-BPSK) performance. Matlab® is used to simulate probability of bit error (P_b) under multiple access interference (MAI) and multipath interference (MPI) conditions.

1.3 Methodology

This research used Matlab® to simulate a complete UWB communication system, including the transmitter, channel, and receiver. Pulse generation, transmission, detection and estimation occur entirely within the software. This permits careful control of all parameters and simple adjustments for future experimentation. The model is validated using theoretical models for antipodal and orthogonal signalling and subsequently extended to include simulated network performance with multipath, multiple access, and narrow band interferers.

1.4 Equipment

Matlab® Version 6.1.0.450 Release 12.1 was used for simulation code development and JMP5.0 for statistical analysis. Hardware consisted of a Dell Personal Computer running Microsoft Windows 2000, Service Pack 2. The computer had 1.0 GB of RAM and used an Intel Xeo 1700 MHz processor. In addition, simulations were run on Linux based platforms having comparable characteristics.

1.5 Thesis Organization

This document is organized into five chapters. This first chapter provides an introduction to ultra wide band (UWB) communications and outlines the thesis document. Chapter 2 provides UWB background information based on relevant literature and previously published results. Chapter 3 describes the methodology used to conduct the research. Results are presented in Chapter 4. Chapter 5 provides conclusions and suggested future work related to this thesis. Two appendices are included that contain additional simulation results and provide the Matlab® code used in the simulations.

II. Background

2.1 UWB History

The typical modern day radio uses a carrier-based modulation technique. However, in the earliest days of wireless communications, only pulsed radios were available. In 1893, Heinrich Hertz began using a pulsed spark discharge for his radio experiments [1]. Modern contributions to the field began in the 1960s, with the efforts of Harmuth at Catholic University of America, Ross and Robins at Sperry Rand Corporation, USAF's Rome Air Development Center, and in Russia [3]. From these early works names such as impulse, carrier-free, non-sinusoidal, and baseband were derived - synonyms for the UWB technology discussed here. Early work was primarily focused on developing radar applications using UWB techniques. Though similar in nature, radar work fundamentally differs from communications applications where power levels are on the order of microwatts (radar power is on the order of kilowatts). Research continued with the first unclassified UWB communications program occurring in 1994 [3].

2.2 UWB Definition

2.2.1 Unlicensed UWB Operation. The sudden explosion of interest in the field of UWB waveforms is due primarily to a recent ruling by the FCC permitting unlicensed UWB operation across the frequency range of 3.1 to 10.6 GHz under rules for Part 15.209 [8]. By establishing this spectral region, the FCC is affording the field of UWB communications room to grow significantly, gaining an advantage over previous forms of data communications that are experiencing spectral over-crowding in their assigned zones.

2.2.2 What is UWB? A UWB radio is a radiator having (1) a bandwidth greater than or equal to 500 MHz, or (2) a fractional bandwidth greater than 20%, where fractional bandwidth B_f is

$$B_f = 2\left(\frac{f_H - f_L}{f_H + f_L}\right) > 0.20 \tag{2.1}$$

where f_L and f_H are frequencies measured at the -10 dB emission points [8]. For the purpose of power emissions, UWB systems fall into one of three categories: imaging, vehicular radar, and communications and measurement. Within the category of communications and measurement, indoor and outdoor systems are treated separately. As of February 2002, the FCC allows operation of indoor UWB systems under the power emission mask given in Fig. 2.1 [8]. The 3.1 to 10.6 GHz frequency range is an unlicensed spectrum where the FCC anticipates most commercial UWB systems will operate. Hand-held outdoor devices have the same power requirements in this 7.5 GHz band with lower emissions requirements elsewhere to avoid interference with existing technologies such as the Global Positioning System (GPS).

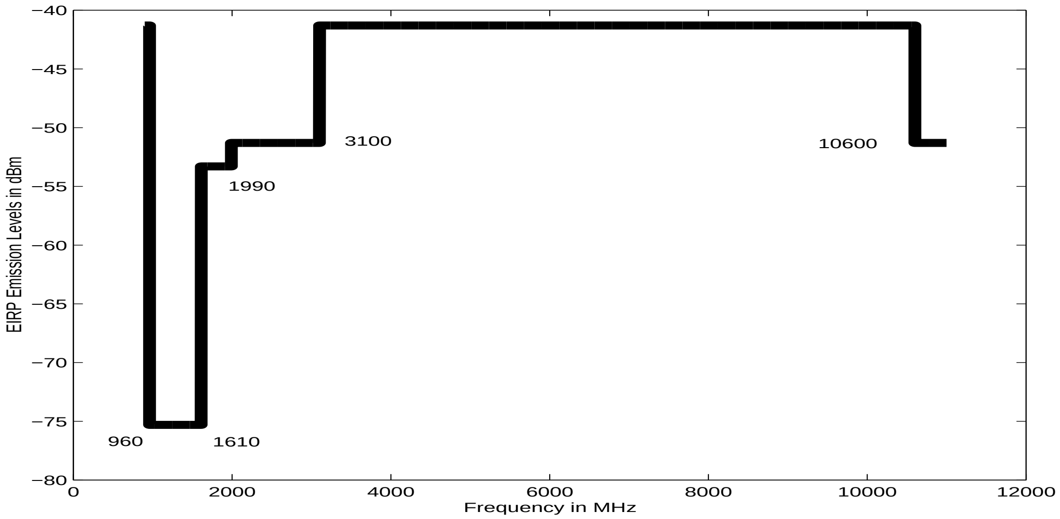


Figure 2.1 Power Emission Limits for Indoor UWB Communication [8]

In the unlicensed spectrum of 3.1 to 10.6 GHz, the Equivalent Isotropic Radiated Power (EIRP) density must remain below -41.3 dBm/MHz. This number is derived from FCC Part 15.209 which dictates that intentional emitting devices must radiate less than 500 $\mu\, V/m$ as measured at a distance of 3.0 meters over a 1.0 MHz bandwidth. The conversion between power (in Watts) and electric field strength is accomplished using

$$P = \frac{E_o^2 4\pi R^2}{\eta} \qquad (Watts) \tag{2.2}$$

where E_o = Electric field strength in V/m
R = Radius of the sphere at which the field strength is measured
η = Characteristic impedance of a vacuum (377 ohms).

2.2.3 Gaussian Impulse Waveforms. Fundamentally, UWB signalling transmits an impulse of energy over a designated period of time (pulse duration T) and repeats the transmission at intervals of T_o (the reciprocal of the pulse repetition rate f_o). The ratio of pulse duration to pulse repetition time is called the duty cycle, i.e., duty cyle=τ=T/T_o.

The form of the transmitted pulse used for UWB applications is typically chosen to be a Gaussian monocycle as illustrated in Figure 2.2. The Gaussian pulse is chosen because it is relatively easy to generate using a step-recovery diode and alternating current. The monocycle is used for analysis because it is the derivative of a Gaussian function, which occurs as a result of the effects of transmission over a UWB antenna. The Gaussian monocycle is well documented in literature [20] and can be analytically represented in the time and frequency domains by (2.3) and (2.4), respectively

$$w(t) = 2\sqrt{e}A\pi t\, f_c\, exp[-2(\pi t\, f_c)^2] \tag{2.3}$$

$$W(f) = -\frac{j}{2}\sqrt{\frac{2e}{\pi}}\frac{A}{f_c^2} f\; exp\left[-\frac{1}{2}\left(\frac{f}{f_c}\right)^2\right] \tag{2.4}$$

where A is the amplitude of the monocycle and f_c is its center frequency having the relationship $f_c = 1/T$.

The second derivative of a Gaussian impulse is used as the received waveform, accounting for the effects of the receiving antenna and is defined as

$$w(t) = \left[1 - 4\pi \left(\frac{t}{\tau_m}\right)^2\right] exp \left[-2\pi \left(\frac{t}{\tau_m}\right)^2\right] \tag{2.5}$$

where τ_m is the impulse width parameter, approximately equal to 0.4 times the pulse width T [24]. To study waveforms in the 5.0 GHz frequency range, pulse durations on the order of $T = 0.2\ ns$ are used. Using codes of length N_c, the PRI is set equal to $T_o = 2T \times N_c$, resulting in typical values of T_o ranging from 10 to 100 ns, depending upon the desired data rate. For UWB pulses having these parameters, the center frequency is approximately 5.0 GHz, as seen in the spectral response of Figure 2.3.

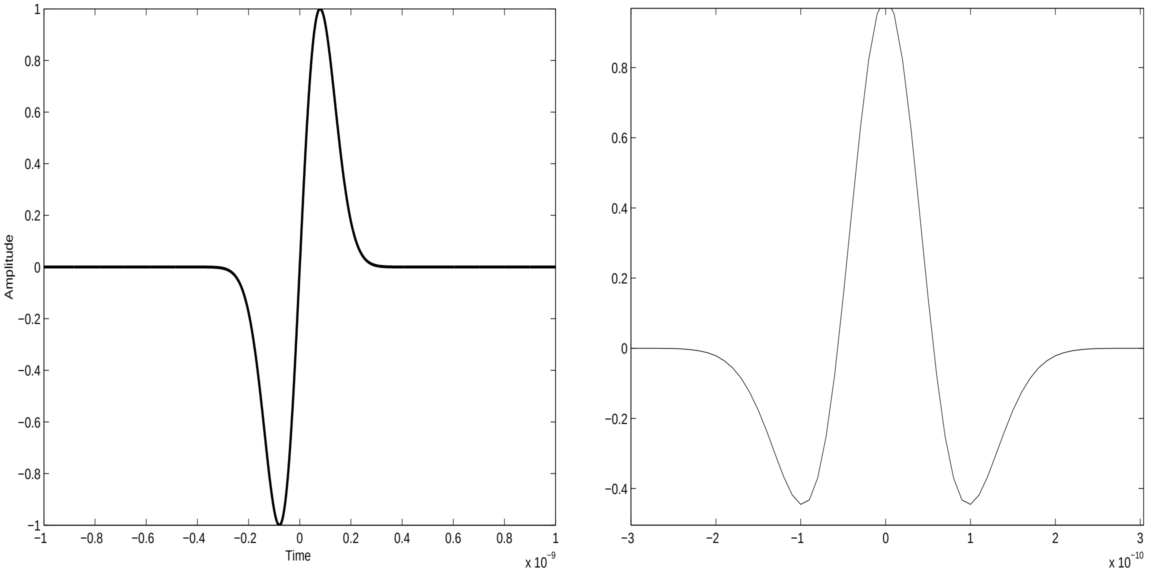


Figure 2.2 Gaussian Impulses: 1^{st} Derivative (Left) and 2^{nd} Derivative (Right)

The transmitted monocycle can be modelled using a periodic pulse train. Using Fourier analysis, the time domain waveform can be represented as a sum of weighted

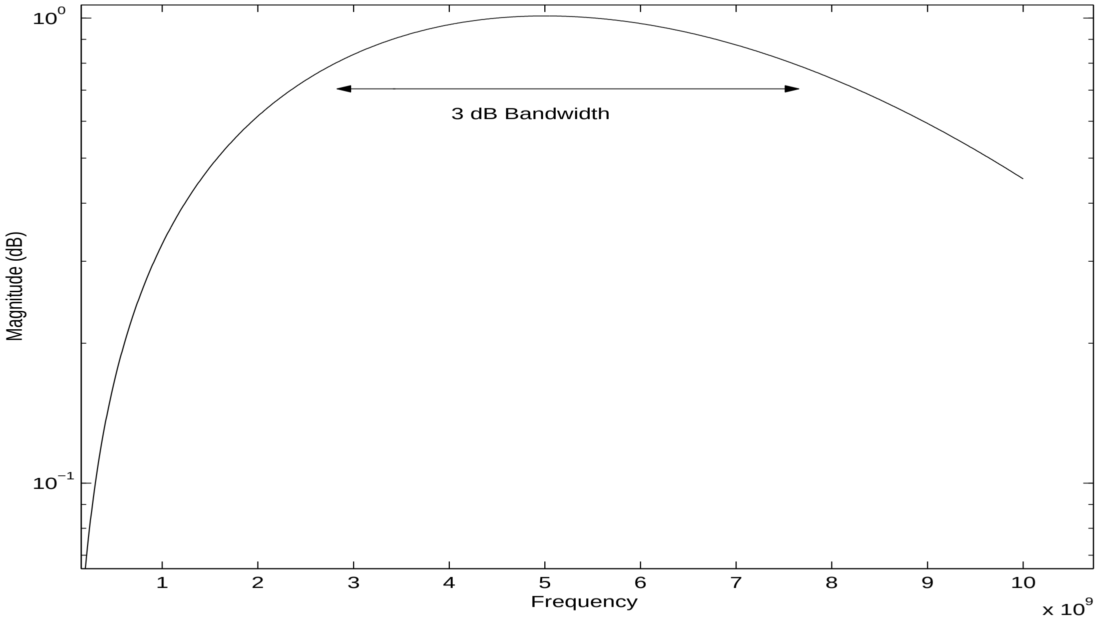


Figure 2.3 Gaussian Monocycle Frequency Domain Response for $f_c = 5GHz$

sinusoids (i.e., Fourier series) with coefficients c_n given by

$$c_n = \frac{1}{T_o} \int_{-T_o/2}^{T_o/2} e^{-j2\pi n f_o t} dt. \tag{2.6}$$

The Fourier series coefficients c_n for a rectangular pulse train are [31]

$$c_n = \frac{AT}{T_o} \frac{sin(\pi nT/T_o)}{\pi nT/T_o} = \frac{AT}{T_o} sinc\left(\frac{nT}{T_o}\right). \tag{2.7}$$

Given a Fourier series representation of

$$x(t) = \sum_{n=-\infty}^{\infty} c_n e^{j2\pi n f_o t} \tag{2.8}$$

the equation for periodic pulse train $x_p(t)$ can now be represented as

$$x_p(t) = \frac{AT}{T_o} \sum_{n=-\infty}^{\infty} sinc\left(\frac{nT}{T_o}\right) e^{j2\pi n f_o t} \tag{2.9}$$

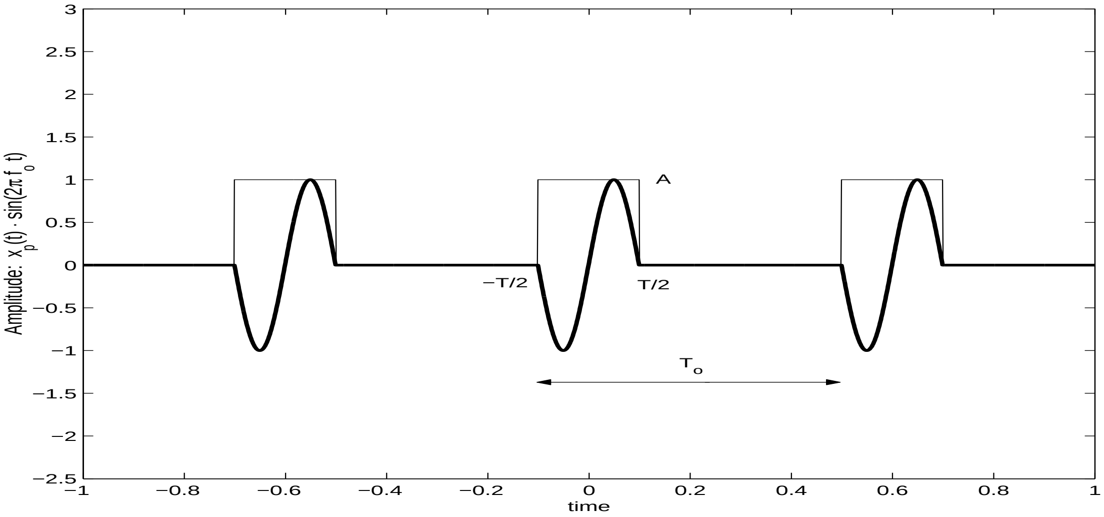


Figure 2.4 Overlay of Pulse Train With Transmitted UWB Waveform

and is shown in Fig. 2.4. In the frequency domain, the periodic signal is a discrete function having a sinc(·) shaped envelope as shown in Fig. 2.5. Note that the resolution of this function, i.e., the spectral line spacing, is determined by pulse repetition frequency $1/T_o$.

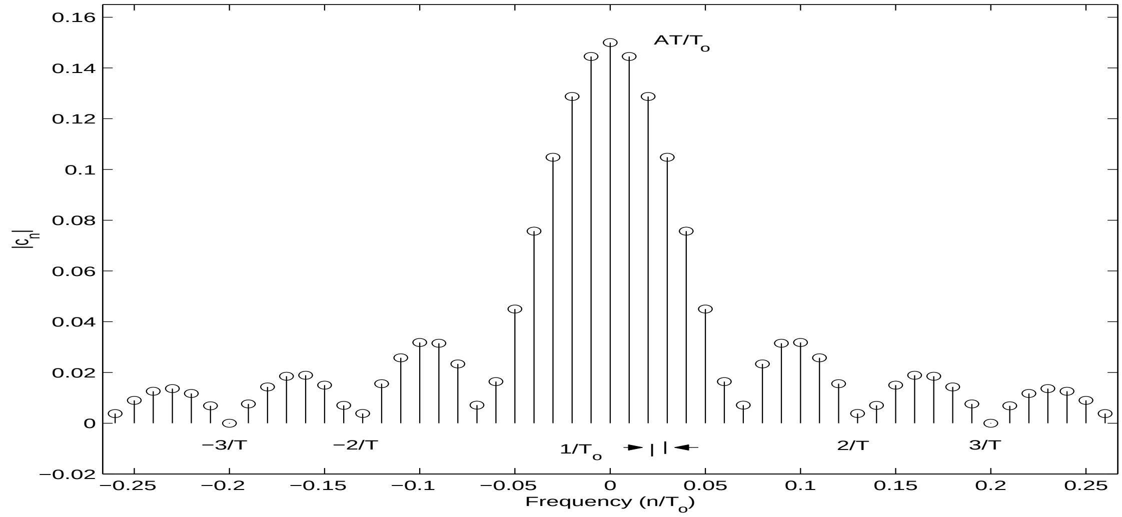


Figure 2.5 Pulse Train Spectrum

Finally, UWB impulses are actually truncated sinusoidal waveforms with a frequency given by the reciprocal of the pulse duration. Using the Frequency Convolution Property of Fourier Transforms [31], multiplication of the pulse train by a carrier wave is equivalent to convolution of the pulse train spectrum with the Fourier Transform of a sinusoidal wave. This effectively shifts the baseband spectrum of Fig. 2.5 to the desired center frequency. Used in this way, the term "carrierless" for UWB is misleading since the UWB spectral response is inherently *centered* at the fundamental frequency of the Gaussian monocycle.

2.3 UWB Approaches and Modulations

Ultra wide band systems can use spread spectrum techniques, along with modulation, to enable multiple access (MA) capability; Time Hopping (TH) and Direct Sequence (DS) are two commonly used methods. In Time Hopping Pulse Position Modulation (TH-PPM), the information in a train of Gaussian pulses is contained in the pulse position relative to the repetition time interval T_o. To allow asynchronous communications and multiple access, a time hopping factor delay is added to offset the position of each signal of user k. A data modulation factor delay is included to shift pulses in a binary PPM scheme.

The second form of multiple access modulation is Direct Sequence Binary Phase Shift Keying (DS-BPSK) [11]. Rather than modulate the signal by a time delay, DS-UWB uses 180^o phase shifts for binary signaling. TH-PPM spreads the signal in time to obtain multiple access. DS-BPSK spreads the signal by multiplying each pulse by a user specific code of amplitude ± 1 and duration T_c. To distinguish users, a distinct spread spectrum code is assigned. In contrast to traditional spread spectrum techniques, when using either the TH-PPM or DS-BPSK signalling scheme, the shape of the pulse waveform does not change.

Though TH-PPM and DS-BPSK appear to be the most common forms of signal modulation in UWB communications, other modulation techniques can be employed. Literature is available on virtually every type of UWB modulation scheme, including: Pulse Position Modulation (PPM), Binary Phase Shift Keying (BPSK) or antipodal signaling, On-Off Keying (OOK), and Pulse Amplitude Modulation (PAM). Combinations of various modulation schemes have also been considered.

2.3.1 Time Hopping and Pulse Position Modulation. Historically, Time Hopping PPM (TH-PPM) has become synonymous with time-modulated UWB. In a TH-PPM pulse train of Gaussian monocycles, signal information is contained in the pulse position relative to repetition time interval T_o. For instance, a pulse arriving at T_o is considered a binary value 0 while a pulse arriving just after the reference time is deemed a 1. A mathematical TH-PPM representation $s^{(k)}[t^{(k)}]$ is [26]

$$s^{(k)}\left[t^{(k)}\right] = \sqrt{N_c P_k} \times \sum_{j=-\infty}^{\infty} w\left[t^{(k)} - jT_o - c_j^{(k)} T_c - d_j^{(k)}\right] \tag{2.10}$$

where P_k = Average power (one code period)
N_c = Spreading code length
$t^{(k)}$ = k^{th} transmitter's clock time
T_c = Spreading code chip period
$c_j^{(k)}$ = Unique time-hopping sequence
$d_j^{(k)}$ = Data modulation sequence

The argument of $w(\cdot)$ in (2.10) contains four timing components. The transmitter's clock time, $t^{(k)}$, represents an arbitrary origin for the k^{th} user. This original pulse reoccurs within every PRI of T_o. For asynchronous communication and multiple access capability, a time hopping factor is added using pseudo-random sequence $c_j^{(k)}$, a uniquely assigned user code. Finally, (2.10) includes a data modulation factor $d_j^{(k)}$ to appropriately shift the pulse position in accordance with the binary PPM modulation scheme.

There are several versions of time-hopping. In [17], a method is presented for a Delay-Hopped Transmitted Reference (DHTR). As shown in Fig. 2.6, a transmitted pair of identical pulses, called a doublet, is separated by time D that is known to the receiver and transmitter. The second pulse is pulse position modulated relative to the first pulse to transmit information. Keeping the first pulse at a fixed interval in time, this scheme reduces errors in transmission by providing a fixed reference to the receiver for making decisions based upon position location.

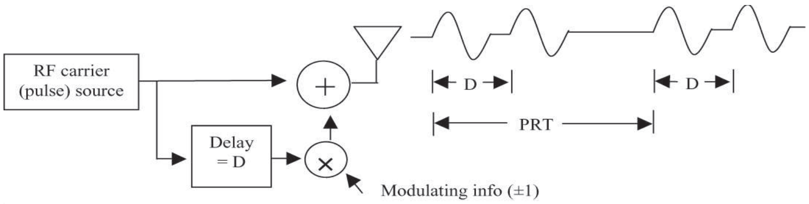


Figure 2.6 Delay-Hopped Transmitted Reference Illustration[17]

Time-hopping is also used in combination with BPSK to create TH-BPSK modulation. In this method, the pulse is pseudo-randomly placed within the pulse interval according to the TH code. Whereas TH-PPM positions the modulated pulse relative to the beginning of a chip, TH-BPSK positions the pulse at the same offset within each chip, but changes the phase by 180^o (a typical form of antipodal signalling). By combining PPM and BPSK, four signal states are obtained within a symbol interval (i.e., 4-Ary signalling).

2.3.2 Direct Sequence and Binary Phase Shift Keying. Similar to conventional Direct Sequence Spread Spectrum (DSSS), DS-UWB has a high duty cycle, phase coded sequence of wide band pulses transmitted at near gigahertz rates. The receiver uses a similar code to convert (de-spread) the signal back to its original data rate. In this way, multiple pulses are encoded to represent one data bit. For a fixed pulse rate, there is an inverse relationship between data rate and signal energy per bit.

The use of DS-BPSK modulation for UWB communications, as described in [11], is illustrated in Fig. 2.7. The DS-UWB technique uses 180^o phase shifts

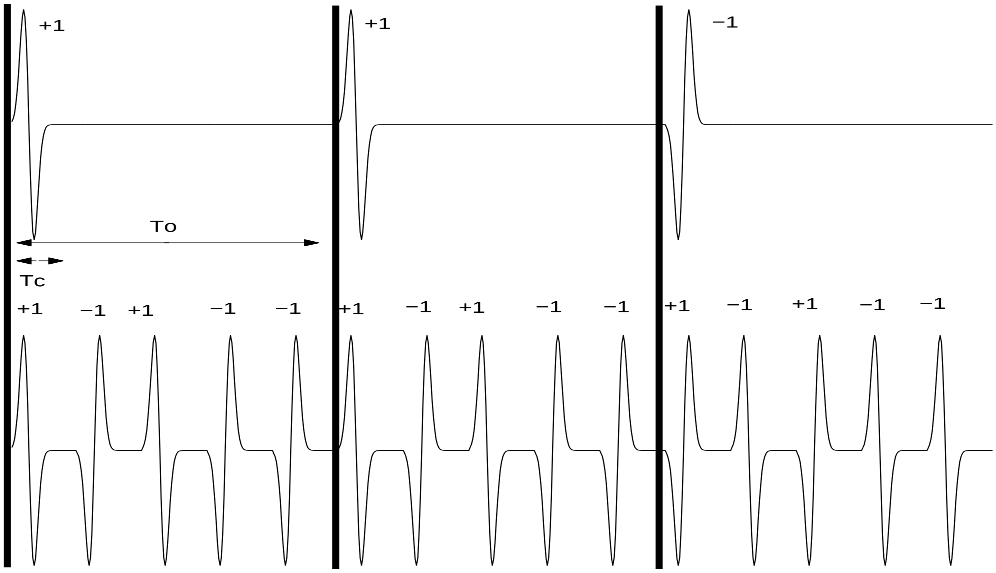


Figure 2.7 Direct Sequence Binary Phase Shift Keying for UWB Illustration

to enable binary signaling as

$$s^{(k)}\left[t^{(k)}\right] = \sqrt{P_k}\sum_{j=-\infty}^{\infty}\sum_{n=0}^{N_{ss}-1} b_j^{(k)} a_n^{(k)} \times w\left[t^{(k)} - jT_o - n^{(k)}T_c\right] \tag{2.11}$$

where $N_{ss} = T_o/T_c$= Spread Spectrum Processing Gain
$b_j^{(k)}$ = Modulated data symbol value for k^{th} user (±1)
$a_n^{(k)}$ = Spreading chip value for k^{th} user (±1).

The DS-BPSK modulation technique spreads the signal across each PRI by multiplying each pulse by a user specific code of amplitude ±1 and duration T_c. To distinguish users, the PRI T_o is an integer multiple of T_c so that multiple pulses are used to represent the multiple chips of the spread spectrum code.

2.3.3 TH and DS Analysis. Several papers have evaluated the strengths and weaknesses of TH-PPM, TH-BPSK, and DS-BPSK [11],[24],[32]. These first findings indicate that DS may reduce the impact of multiuser interference but time-hopping is more efficient at reducing multipath and narrow band interference effects. For a simple matched filter receiver, it is claimed that bipolar modulation performs better than TH-PPM. In [32], it is stated that DS-BPSK is more suitable for high data rate systems since it can accommodate higher PRF values than time-hopping codes. Fundamentally, TH modulation represents a form of orthogonal signalling and DS modulation is a form of antipodal signalling. Consequently, it is necessary to assess the communication performance of the two techniques independently and great care must be taken when drawing conclusions.

2.3.4 Differences between UWB and DSSS. To understand how UWB signalling differs from traditional DSSS, it is insightful to consider a simple comparison between them. In typical communication models, Additive White Gaussian Noise (AWGN) is assumed present in the channel. Since average noise power (N_{av}) is a function of bandwidth, the same N_{av} is not present in a narrow band receiver as present in a UWB receiver. Consequently, care must be taken when evaluating performance in terms of power-based metrics such as bit energy per noise power E_b/N_o.

In all UWB signalling, the average energy per bit E_b is a function of bandwidth and thus affected by pulse repetition interval T_o and the peak transmit power. To preserve average energy, peak power is adjusted for changes in T_o. On the other hand, E_b in a DSSS system is dependent on bandwidth before spreading so the energy is a function of the data rate not the chip rate. Consequently, it is difficult to directly compare the bandwidth of a UWB system and that of a DSSS system in terms of energy.

Finally, figures of merit have been developed to compare DSSS and UWB in terms of "processing gain" with varying levels of accuracy. For DSSS systems, processing gain ($G_p = R_c/R_d$) is a comparison of chip rate to data rate. The equivalent to the number of chips per data bit in DSSS is the number of integrated pulses in

UWB. For UWB, “processing gain” is simply the number of coherent pulses used to represent a single bit of information. In both cases, data rate is traded for an increase in SNR. If both SNR and data rate are required to increase, then both systems must increase in complexity and power consumption. The limiting component of a UWB system is the minimum T_o that can be achieved to keep the peak power under specified emission levels while maintaining average transmission power. In contrast, the limiting component of a DSSS system is the complexity of the receiver to maintain a chipping rate R_c. In this regard, above a certain threshold of required processing gain, UWB may offer a simple alternative. Given most commercial DSSS systems operate between 902-928 MHz, 2400-2483.5 MHz, and 5725-5850 MHz, the recent UWB spectrum allocation offers an opportunity for growth that DSSS may not provide [13].

2.3.5 Timing Jitter.

When using a periodic pulse train at fixed intervals, the power spectrum exhibits a “comb-like” response. To make UWB transmissions appear more noise-like, thereby decreasing interference and unwanted detection, spectral nulls can be displaced throughout the spectrum. Pulse position modulation is one way by which the pulse interval may be varied slightly to spread the spectrum power. In addition, pseudo-random time-hopping codes are implemented to dither the pulse location inside of each pulse interval as shown in (2.10). Using this method of timing jitter, the power spectral density for a randomly jittered pulse train is [12]

$$S_x(f) = [T\,A\,sinc(f\,T)]^2 \left[\frac{sinc^2(2f\beta)}{4T_o^2} \sum_{n=-\infty}^{\infty} \delta(f - \frac{n}{T_o}) + \frac{2 - sinc^2(2f\beta)}{4T_o} \right] \qquad (2.12)$$

where pulse jitter is uniformly distributed over the interval $[-\beta, +\beta]$. The power spectral density plots of Fig. 2.8 indicate the effect dithering can have on interfering discrete frequency components. The discrete and continuous frequencies are illustrated as dotted and solid lines, respectively. In the small β case, pulse dithering occurs uniformly over 10% of T_o. When dithering is allowed to occur over 90% of T_o, there is a 20 dB reduction in the sidelobes of the discrete frequencies, i.e., the dotted lines are attenuated by 20 dB. No amount of dithering can completely remove the discrete spectral interferers. Current UWB technology allows time dithering, using pseudo-noise codes, to typically place a pulse within a 3 picosecond accuracy.

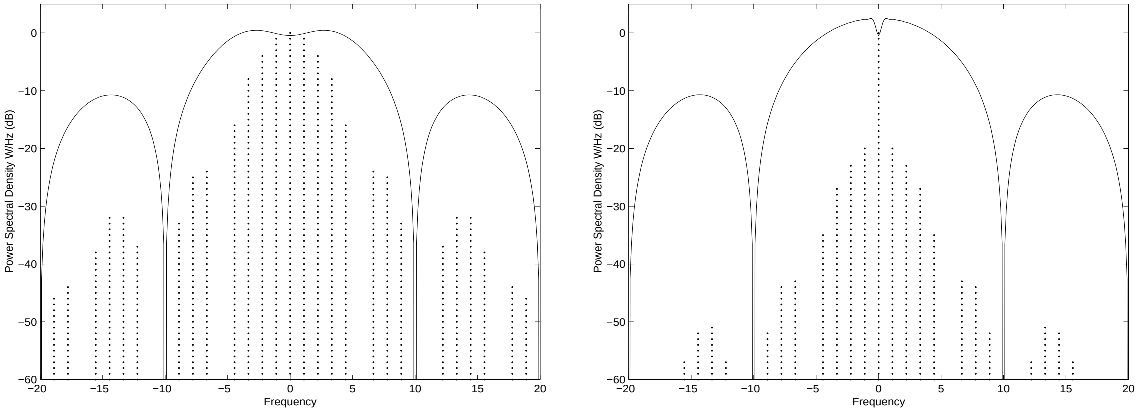


Figure 2.8 Power Spectral Density of Pulses with Timing Jitter: Small Beta (Left) and Large Beta (Right)

2.4 Short Pulse Characteristics

Ultra wide band techniques have several distinct characteristics revealed by comparison to other types of signalling. As the name implies, UWB techniques generate signals of large bandwidth using a very narrow transmitted pulse. Due to the large waveform bandwidth, a comparison to the Shannon Capacity is in order. The Shannon capacity C of a system [27] is

$$C = W \times log_2(1 + SNR) \quad \left(\frac{bits}{sec}\right) \tag{2.13}$$

where W is the bandwidth in Hertz. UWB systems maximize data capacity through increased bandwidth without an increase in signal-to-noise ratio (SNR).

UWB has unique properties due to its short pulse nature in time: resistance to interference due to its ability to resolve multipath propagation issues and low power requirements due to a low duty cycle. As demonstrated in Fig. 2.9, the path that a

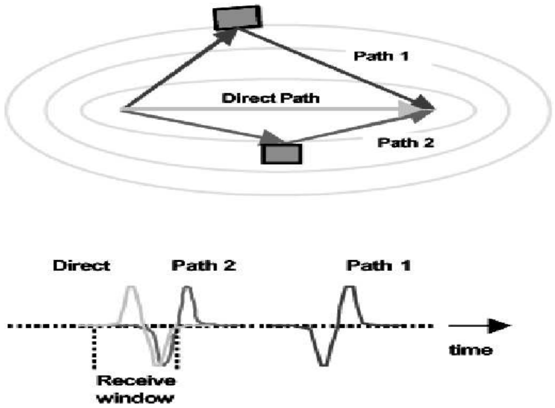


Figure 2.9 Illustration of Multipath Effects [30]

signal takes to reach the destination receiver can greatly impact the recognition of distinct pulses sent. As the number of pulses and the variance in transmission delay increases, pulses begin to overlap so that constructive and destructive interference occurs. Since UWB systems transmit sub-nanosecond pulses, they are able to resolve overlapping signals with much higher resolution than technologies using higher pulse durations. Subsequently, since the fundamental law of physics requires that distance $d = c \times t =$ speed of light × time, for a pulse travelling in free-space, a distance of $(3 \times 10^8) \times 1$ ns $= .3$m $\approx$ 1ft is travelled every nanosecond. By increasing time resolution to overcome the effects of multipath distortion, UWB increases signal location resolution which results in increased range accuracy, or precision location. This makes UWB technology valuable for indoor, high clutter communications.

The use of UWB signalling removes the necessity for heterodyning. As developed by Howard Armstrong in 1918, and used in typical carrier-based modulation schemes, heterodyning provides frequency translation for typical narrow band radios or wireless communication schemes such as Bluetooth and IEEE 802.11 [9],[19]. At low frequencies, it is relatively easy to provide stable circuitry for signal demodulation and data recovery. The increased need for higher data rates dictates that operating frequencies increase. For systems operating at higher frequencies, such as that of the cellular telephony industry, it is common to require two or three stages

Table 2.1 Spatial Capacity Values for Various Wireless Technologies

Technology	Data Rate	Systems/Area	Radius of Area	Spatial Capacity
IEEE802.11b	11 Mbps	3	100 meters	1,000 $\frac{b}{s \cdot m^2}$
Bluetooth	1 Mbps	10	10 meters	30,000 $\frac{b}{s \cdot m^2}$
IEEE802.11a	54 Mbps	12	50 meters	83,000 $\frac{b}{s \cdot m^2}$
UWB	50 Mbps	6	10 meters	1,000,000 $\frac{b}{s \cdot m^2}$

of conversion to move the signal to acceptable levels for filtering and information recovery. Avoiding these intermediate frequency steps helps UWB technology to remain simple and cost effective.

To compare the data rate per unit of area of physical coverage, the metric of spatial capacity, or spatial efficiency, has been introduced [9] and is defined in units of *(bits/sec)/square-meter*. In using this metric, improved performance comes from the ability to fit more units, capable of transmitting at higher data rates, in smaller spaces. The numbers listed in Table 2.1 are approximate values for typical systems using the respective wireless technology. Using this metric, it is clear that UWB systems can transmit orders-of-magnitude more data in a given space. Spatial capacity is biased toward methods that transmit shorter distances so that more devices can transmit in the same area. This metric is less meaningful for comparing technologies when large propagation distances are desirable or the full capacity of devices is not available.

It is important to understand the trade-offs when varying UWB waveform duty cycle - the ratio of pulse length T to symbol duration T_o. When determining the average SNR for a UWB system, the symbol energy can be thought of as the average power multiplied by the pulse repetition interval, yielding the relationship

$$E_s/N_o = P_{avg}T_o/N_o = [P_{SD}/N_o] \cdot [f/f_o] \tag{2.14}$$

where $f_o = 1/T_o =$ Pulse repetition rate (prf)

$f = 1/T =$ Bandwidth of the transmitted pulse

P_{SD} = Average Signal Power Spectral Density

N_o = Noise power spectral density.

Average transmitted power is given by $P_{avg} = f\,P_{SD}$ with P_{SD} limits set by FCC power emissions regulations at -41.3 dBm/MHz. From (2.14), it is evident that decreasing the pulse duration or increasing the pulse repetition interval (equivalent to increasing the pulse bandwidth or decreasing the pulse repetition frequency) enables communications across greater distances for a fixed average signal power and spectral density. In light of the relationship between f/f_o and the effect it has on system power, some literature refers to this quantity as "pulse processing gain" [9]. By varying the UWB waveform duty cycle, this "gain" maintains average power at the expense of increased peak power or much lower data rates.

2.4.1 Interference Concerns. As with any new technology, the use of UWB signalling introduces concerns regarding coexistence with existing technologies as well as the practicality of implementation. During the past few years, hundreds of studies have begun to assess reliable channel models, effective receiver designs, precise pulse-shaping techniques, and efficient multi-user access schemes for UWB applications.

A primary concern with UWB is coexistence with narrow band receivers and the mutual interference induced. It is instructive to calculate noise floor power and received UWB interfering signal power P_I at a narrow band receiver. In the case of a 20 MHz receiver designed for operation with a IEEE 802.11 wireless system, a typical noise power P_N may be

$$P_N = KT_oB = -101.0\ dBm \qquad (2.15)$$

where K = $1.38{\cdot}10^{-23} J/^oK$

$T_o = 290^oK$

B = 20 MHz.

Using (2.16) [31], the received UWB interfering power P_I in a 20 MHz receiver can be calculated at a distance of $d = 8.0$ meters from the UWB source using an antenna with a 6.0 dB gain. Using FCC Part 15.209 specifications for UWB power emissions, UWB systems can isotropically radiate (EIRP) at -41.3dBm/MHz in the 3.1 to 10.6 GHz range. To determine P_I, a frequency dependent wavelength value must be chosen. For illustration, a value of 5.0 GHz is arbitrarily chosen for the frequency. Since the allowable UWB bandwidth is approximately 7.5 GHz at these emission levels, the actual interfering power may vary by several decibels depending upon the frequency (-82.6 dBm at f_c =3.1 GHz to -93.3 dBm at f_c =10.6 GHz). The received UWB interfering power is

$$P_I = \frac{EIRP \cdot G_r \lambda^2}{(4\pi d)^2} = \frac{EIRP \cdot G_r}{L_s} \tag{2.16}$$

where

$$EIRP = 10^{-4.13dBm/MHz} \cdot 20MHz = -28.3 \text{ dBm}$$

$$G_r = 6.0 \text{ dB}$$

$$\text{d} = 8.0 \text{ meters}$$

$$\lambda = c/f = \tfrac{3 \cdot 10^8 m/s}{5 \cdot 10^9 GHz} = .06 \text{ m}$$

$$L_s = \tfrac{\lambda^2}{(4\pi d)^2} = 64.5 \text{ dB.}$$

The value L_s is the path loss. Assuming only free-space loss, the received power calculation yields $P_r = EIRP + G_r - L_s = -86.8$ dBm. This represents an increase in the effective interference plus noise floor level of about 15 dB for a 20 MHz narrow band receiver at $d = 8.0$ meters from a single UWB device operating at maximum allowable emission level. Of course, if the path loss is varied to account for increased fading due to walls and other medium typically observed in an 8.0 meter radius indoors, the interference level may significantly change. Research indicates that the use of more accurate fading channel models with a Rayleigh distribution for path loss results in the UWB radio imposing insignificant amounts of interference on a narrow band receiver at distances of at least one meter [29].

The aggregate effects of multiple UWB transmissions in overlapping coverage zones is another cause for concern. The NTIA [22] characterized this effect, and experimentation to determine a reliable model is on-going. Research presented in subsequent chapters of this document addresses some of the multiple transmitter scenarios.

Methods for implementing UWB technology are continuously improving. Precise pulse-shaping, filter matching, and improved synchronization are all issues. Analysis is underway to determine the best receiver type for UWB signalling. Current UWB systems typically use a RAKE receiver architecture, in which a predetermined number of received signal correlator outputs are averaged to increase signal energy detection in multipath situations. This averaging of SNR values over a given symbol duration permits more accurate location and phase determination of a UWB transmitted pulse.

2.4.2 Overspreading and Channel Fading. The ability of UWB techniques to resolve multipath issues is of great significance when determining how to best model and overcome fading channel effects. Since transmitted pulse duration determines communication system bandwidth and multipath signal resolution, it would seem that the smallest pulse duration should yield the best performance. The ability to resolve individual paths would then allow for the maximum amount of system users. However, as pulse duration is reduced, the total energy in the time allotted for pulse recognition due to a single path - commonly called a bin - is reduced. The point at which the energy per bin, and thus the signal SNR, drops below a level necessary to communicate over an arbitrary distance is called overspreading [33].

The most widely used parameter for measuring the channel delay spread is the root-mean-square (RMS) delay spread. Typical values for RMS delay spread for an indoor channel are between 15 to 30 nanoseconds for antenna separations of 5 to 30 meters [10],[11]. To account for amplitude fading that occurs over these delays, several models have been proposed for path loss and channel fading characterizations. UWB performance analysis often assumes a Rayleigh fading distribution because a large number of multipath components are assumed to occur in the same bin a priori. Since UWB signal resolution enables determination of distinct multipath components, alternative distributions to the exponential amplitude loss associated with Rayleigh fading have also been used [5]. A log-normal distribution for multipath amplitude fading is one such proposed model. To model multipath arrival times, a $\triangle - K$ model is sometimes used [10]. This model uses a modified Poisson process to incorporate a two-state Markov model for path arrival probabilities. In this way, the arrival rate is a function of whether an arrival occurred in the previous time slot and is a function of empirically based measurements for the probability of path occurrence at various delays.

2.5 Coding Techniques and Metrics

Development of UWB MA communications favors codes having low cross-correlation to minimize MAI and low auto-correlation sidelobes to minimize multipath interference. The notable metrics used to characterize these properties are the Peak Sidelobe Level (PSL), Integrated Sidelobe Level (ISL), and Peak Cross-Correlation Level (PCCL).

2.5.1 Coding Metrics.

2.5.1.1 Peak Sidelobe Level. The *Peak Sidelobe Level* (PSL) metric is commonly presented as the maximum sidelobe amplitude normalized by the peak autocorrelation response, expressed in decibels. Calculation of PSL is straightforward for biphase codes and begins by first computing the autocorrelation function. A given phase code, C, of length N_c, has a periodic autocorrelation function, X, of length $L = 2N_c - 1$, and can be expressed as [21]

$$X(l) = \sum_{k=1}^{N_c} C_k C_{k+l} \tag{2.17}$$

where index l steps through the range of $[-(N_c - 1) \leq l \leq (N_c - 1)]$. The peak autocorrelation response occurs at the mid-point, $X(l = 0)$, and has amplitude N_c. The method for representing the code's PSL permits direct comparisons of codes having different lengths by expressing the normalized PSL value, in decibels, as referenced to the peak autocorrelation response [21]

$$PSL(dB) = 10 \log \left[\max \left(\frac{X_l^2}{N_c^2} \right) \right] \qquad l \neq 0. \tag{2.18}$$

2.5.1.2 Integrated Sidelobe Level.

Similar to the PSL metric, the *Integrated Sidelobe Level* (ISL) is calculated using the code's autocorrelation function $X(l)$. ISL is a measure of total power present in the sidelobe responses as compared to the power contained in the autocorrelation central peak [21]. ISL is generally expressed in decibels and given by

$$ISL(dB) = 10 \log \left[\sum_{l \neq 0} \frac{X_l^2}{N_c^2} \right]. \tag{2.19}$$

2.5.1.3 Peak Cross-Correlation Level.

The *Peak Cross Correlation Level* (PCCL) is similar to PSL with the exception that the PCCL is determined from the cross-correlation between codes within a given code family [21].

2.5.2 Biphase Codes.

Various spreading codes are used in UWB systems to account for the effects of auto- and cross-correlation characteristics. Though fundamental UWB principles do not exclude the use of polyphase coding, typical UWB systems use biphase codes for antipodal or pulse-position binary modulation schemes.

Published results have used 31 and 63 length Gold codes [11]. Gold code advantages include the number of available codes which are easily generated and their well-defined three-value periodic cross-correlation [25].

A Random Integer (RI) code selection process is implemented using a random binary generator to create binary strings of various lengths. Random integer codes are valuable for code performance comparison since no attempt is made to define the code strings based on achieving given sidelobe level characteristics.

A third code generation process, Simulated Annealing (SA), was first introduced in 1983, with the principle application being optimal computer design. The SA process is based on the concept of molten metal seeking a lower energy ground state while cooling. Similarly, the aperiodic auto- and cross-correlation sidelobe levels form an energy state that can be minimized. The energy expression used for SA code generation is

$$E = w_1 C_a + w_2 C_c \tag{2.20}$$

where C_a and C_c are desired maximum auto- and cross-correlation levels, respectively, and the weights w_1 (auto) and w_2 (cross) are used to assign relative importance to correlation levels during the minimization process. The SA code generation process was demonstrated in [2] and produces binary and polyphase codes of arbitrary length according to user defined correlation weighting.

III. Methodology

3.1 Problem Definition

This research characterizes bit error performance (P_b) of TH-PPM and DS-BPSK multiple access schemes for UWB communications by first validating communication performance. This is done by varying the Signal-to-Noise Ratio (SNR), which proportionally maps to E_b/N_o, or energy per bit divided by Additive White Gaussian Noise (AWGN) power spectral density. The ratio of incorrectly estimated bits to total number of bits received is used to calculate P_b. Communication performance under interference conditions of Multiple Access Interference (MAI) and Multipath Interference (MPI) is simulated to study the effects of UWB communication systems operating in a network of users with realistic propagation delays. In addition, modelling is done to study the effects of Narrow Band Interference (NBI) on UWB receiver performance.

Analysis of TH-PPM and DS-BPSK techniques begins with development of components used in a UWB communication system. The system is tested by introducing AWGN into the channel to validate the model. Various values of noise power are used to test the two multiple access methods. Finally, a specific SNR (E_b/N_o) is chosen to validate against previous results and serve as an appropriate power level for digital communications. Communication performance of TH-PPM and DS-BPSK techniques is assessed, over a range of user capacities, in terms of bit error rate (BER). MAI and MPI are introduced into each scheme and performance results analyzed for both synchronous and asynchronous network operation.

3.2 System Components and Limitations

To study the communication performance of TH-PPM and DS-BPSK modulation schemes, a model is developed and validated using previously published results. Figure 3.1 illustrates a typical UWB communication system.

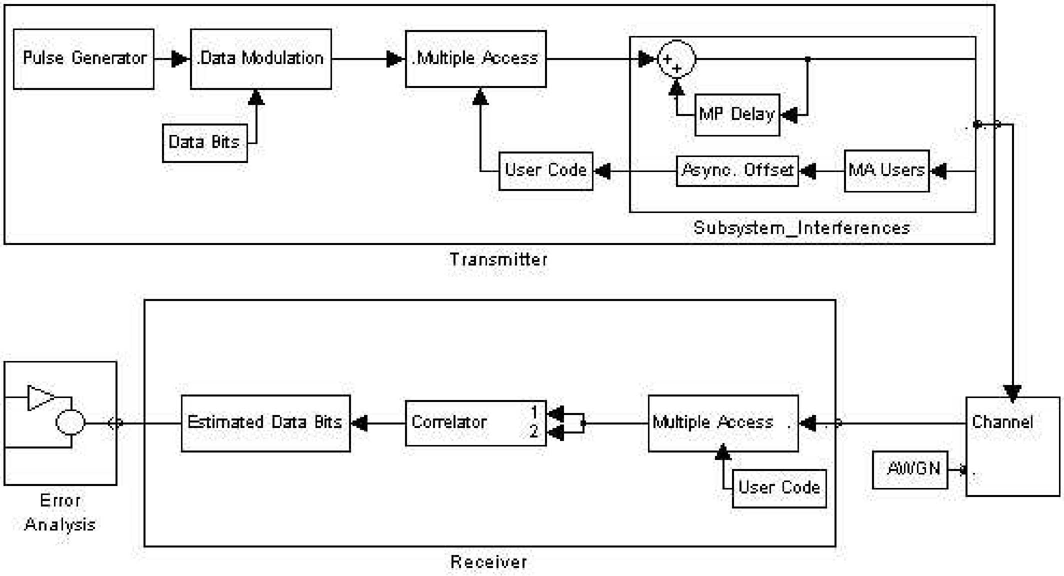


Figure 3.1 Block Diagram of UWB Communication System

The system under test consists of three components: the transmitter, channel, and receiver. Within the transmitter, a pulse generator creates the desired UWB Gaussian waveform. The Gaussian symbol passes into the modulator where Binary Phase Shift Keying (BPSK) or Pulse Position Modulation (PPM) is induced before waveform coding by the appropriate multiple access scheme, i.e., Direct Sequence (DS) or Time Hopping (TH), respectively. For simulations involving interference, MPI and MAI are added following MA signal generation. MPI is inserted by superposing replicated copies of the original waveform for each user. MAI is simulated by incorporating a distinct user code for each user and then transmitting the superposition of all user signals. Separate trials are performed to assess performance in synchronous and asynchronous networks.

Upon transmission, the signal is sent through the channel model, with power adjustments made to control the SNR.

The receiver uses multiple access and demodulation components in reverse order of the transmitter. In addition, the receiver uses a correlation and decision-making stage to intelligibly decipher the received waveform. The two channel correlation receiver tracks the phase of the DS-BPSK signal and the pulse position of the TH-PPM signal. Since the approximate position relative to the beginning of a frame is known in DS-BPSK, the receiver only correlates over pulse duration T to maximize the received SNR, thereby eliminating extra AWGN passing through the correlation filter. In TH-PPM, once the received time hopped waveform has been de-hopped, the received pulse exists in one of two locations (binary signaling) relative to the time hopped position. Therefore, each channel of the correlation receiver integrates over a duration of T to estimate pulse position.

For this study, the components under test are the modulation and multiple access blocks of the TH-PPM and DS-BPSK access schemes. The MA blocks use components to allow for time hopping or direct sequence user distinction and contain the spreading code generators. The performance of these modulation and MA systems, when configured to use various spreading codes, is assessed.

Several limitations have been imposed on the study of this system. The Gaussian waveform generated is fixed in duration and repetition. No selective fading distributions, such as Rayleigh or Rician, are used in the channel. The receiver is comprised of a two channel correlator without any RAKE receiver processing. All power is assumed constant and evenly distributed for the generated signals, including the interference signals. No interleaving or encryption of data is used, other than the multi-user coding. Work has shown successful PPM can be performed using the low sidelobe auto-correlation property of a Gaussian pulse to overlap the pulses in time [34]. For simplicity, this study separates binary pulses of length T by placing each pulse of a given user in a non-overlapping chip of length $2T$.

In accordance with receiver integration limitations addressed in [13], and to permit fair evaluation of the two modulation techniques, processing gain issues are independently considered for each modulation technique when establishing SNR and interference levels. A fair evaluation of TH and DS is accomplished using a fixed *average* signal power in (2.10) and (2.11) when characterizing performance. Since DS provides an inherent processing gain N_{ss} which is N_{ss} times greater than TH (without pulse integration), simulations are performed using a received TH-PPM SNR equaling N_{ss} times the received DS-BPSK SNR within each PRI; this maintains identical demodulator input power for both modulation techniques. Although not a requirement, one period of the $N_c = 31$ bit spreading code is contained within each PRI of T_o for this work such that $N_{ss} = N_c$.

3.3 System Services

The purpose of a wireless communication system is the transmission of symbols across a wireless channel. Bits are digitally created, converted to an analog electromagnetic waveform, transmitted across a noisy lossy channel, received at a destination, sampled and converted back to digital form. These samples are sent through a decision stage that interprets the information and outputs a bit string. In performing this service of transferring data, the outcome for each symbol can be either a correct or incorrect interpretation of the transmitted symbol.

3.4 Performance Metrics

Following pulse generation, data modulation, signal transmission, and reception, the communication symbols are demodulated and the total number of bit errors recorded. All simulations are terminated when the total number of bit errors e_b exceeds 300. Estimated bit error performance is calculated using $P_b = e_b/n$ where n is the total number of transmitted bits required to achieve 300 errors (1 trial). The confidence interval for P_b is [18]

$$r = z\sqrt{\frac{P_b(1 - P_b)}{n}} \tag{3.1}$$

where r is the accuracy of the mean and $z = 1.96$ to establish a 95% confidence interval. Based on (3.1), the worst case simulation results presented in Chapter 4 are within $\pm\, r/P_b = \pm\, 11\%$ of the actual mean with a 95% confidence. This accuracy permits reliable data comparison while sufficiently limiting the computational load of the simulations. The number of pulses used for a given trial ranges from a few hundred pulses to hundreds of millions, increasing in number as the SNR increases.

The performance of time hopping and direct sequence UWB systems is reported using probability of bit error P_b versus SNR. This is a common technique that allows comparison to published results. P_b is a measure of the probability of successful data transmission. SNR is one of the primary parameters studied in communications and provides an indication of signal versus noise power. This becomes significant since power is a limited resource and determines many qualities of a system such as signal propagation distance.

For a given SNR, P_b is compared to the number of users N_u. MAI is related to N_u due to interference caused by the superposition of multiple signals. Using this relationship, the expected BER, for a given SNR, can be determined from the number of simultaneous users supported.

3.5 Parameters

Several parameters control operation of a UWB system. The number of multiple access users is the basis for this work and is certainly a significant system parameter. In generating the waveform, the pulse duration T and repetition interval T_o control the frequency range over which the system operates. For example, a pulse gated on every $T = 0.2ns$ spectrally occurs centered at 5.0 GHz. Pulse duration must be closely controlled since the center frequency is inversely proportional. A small error in pulse width can move system operation outside the bandwidth of receiver filters. Chip time T_c controls the number of pulses used to represent a single bit. Therefore, T_o is set to an integer multiple of T_c.

Table 3.1 System Parameters

Parameter	Symbol	Value
Pulse duration	T	$0.2 \times 10^{-9}s$
Pulse width parameter	τ_m	$0.4 \times T = 0.8 \times 10^{-10}s$
Pulse Repetition Interval (PRI)	T_o	$12.4 \times 10^{-9}s, 12.8 \times 10^{-9}s$ (DS,TH)
A/D sampling resolution	dt	$8.0 \times 10^{-12}s$
Chip duration	T_c	$2 \times T = 0.4 \times 10^{-9}s$

Varying these parameters affects achievable system data rate. For the system tested, the data rates for DS and TH are $1/T_o = 80.645$ Mbps and 78.125 Mbps, respectively. The slight difference in the PRI for the two modulation techniques is due to the code assignment implementation. In DS implementation, a code of length N_c uses N_c pulses of width T_c to represent one bit. This same length code is created from a shift register of length b such that in TH implementation, a bit occurs in one of 2^b possible positions. For example, a code of length $N_c = 31$ requires $31 \times T_c$ seconds to transmit one bit in DS. However, in TH, $2^{(b=5)} = 32 \times T_c$ seconds are required to transmit one bit. Thus, a difference of one chip duration for the PRI of DS and TH is required for all simulations.

Since the system is digital, the sampling rate affects the accuracy of the results. The system must be sampled above the Nyquist rate to minimize aliasing effects. The conversion factor used for mapping between SNR and E_b/N_o is

$$\frac{E_b}{N_o} = \frac{S\,T_s/k}{N\,2/W} = \left(\frac{S}{N}\right)\,\frac{T_s}{k\,\Delta t\,2} \tag{3.2}$$

where T_s is the symbol duration or PRI T_o, $W = 1/\Delta t$ is the signal bandwidth, Δt is the sample spacing, and k is the number of bits per symbol ($k = 1$ for binary modulation).

System workload is a function of the total number of pulses transmitted, the SNR (which determines E_b/N_o), the number of system users, and the number of multipath replications. The probability of bit error is influenced by each of these values. In an ideal system, perfect transmission occurs with sufficiently high SNR, otherwise errors are possible. The total number of pulses must be studied to ensure accurate representation of system performance. Since the number of pulses does vary, the workload of the system changes, though it is not a factor of interest itself.

3.6 Factors

3.6.1 Codes. Variable parameters are known as factors. The multiple access code factor used in this research has six levels, each representing a unique code family. A common code family used in published literature is Gold codes [11]. For baseline performance comparisons, $N_c = 31$ length Gold codes are generated [23]. This research expands previous results by testing a Random Integer (RI) code generation process and four Simulated Annealing (SA) codes of length 31.

A Random Integer (RI) code selection process is implemented using a random binary generator to create binary strings of length 31 for DS modulation. For TH modulation, these codes are mapped to a decimal equivalent time position ranging over [0:31] using a sliding window of $b = 5$ code elements and single element shifts, creating a periodic time hopping code of length $N_c = 31$.

The SA code generation process was demonstrated in [2] and produces binary and polyphase codes of arbitrary length according to user defined correlation weighting. For this work, only biphase codes are considered since the UWB modulation techniques of interest represent binary signaling. SA codes of $N_c = 31$ bits are generated for comparison with equivalent length Gold and RI codes; weights w_1 and w_2 (cf., (2.20)) are set for three cases: to minimize peak cross-correlation *SAC*, to minimize peak auto-correlation *SAA*, and to equally weight the auto- and cross-correlation peaks *SAAC*. A fourth SA code, *SACmod*, is created by rearranging the code order of the SAC code (first five codes are moved in order to the last five).

System performance is highly dependent upon code choice since the success of the correlation receiver depends upon the cross- and auto-correlation characteristics of the codes used. In other words, the probability of bit error is controlled by the number of signal collisions causing the receiver to incorrectly estimate a modulated signal. Since the various codes control pulse position and phase alignment, the choice of user assigned codes is a significant factor in MA and multipath communication performance.

3.6.2 SNR. The E_b/N_o levels range from 0 to +10 dB in increments of 1.0 dB. These values are chosen because they cover the range of interest for typical communications systems with enough detail to draw conclusions and interpolate plots. SNR is varied when studying single user communication performance for model validation and when assessing UWB system performance in narrow band interference. To isolate the effects of multiple users, SNR is not varied during MAI simulations. Instead, the SNR is fixed at $E_b/N_o = 10$ dB. This value is chosen to permit performance comparison with previously published results [11].

3.6.3 MAI. Using a fixed SNR level, the number of simultaneously transmitting users is varied from 1 to 15 to permit comparison with and extension of previously published results. It is expected that as the number of users increases, the BER will increase due to the increased interference caused by each user. This research will quantify that increase.

3.6.4 Synchronization. For a given communication link, the transmitter and receiver are synchronized. However, as another factor, a given user may transmit either synchronously or asynchronously relative to other users. Both synchronous and asynchronous networks are studied here. In TH-PPM, each user is offset in the range of $[0 : T_c]$ relative to other users. For DS-BPSK, the offset occurs anywhere within $[0 : T_c \times (N_c - 1)]$ since N_c chips of width T_c occur within the DS symbol duration.

3.6.5 MPI. Multipath interference (MPI) effects are characterized using an RMS time delay of 15.4 *ns* for each user's replicated signal [11]. Data is generated at an $E_b/N_o = 10$ dB for five levels of multipath replication (N_{MP}) including $N_{MP} = 0, 2, 5, 10$ and 40 multipath replications per user. Simulating MPI provides a method for testing the effectiveness of auto-correlation metrics to characterize code performance since the BER is influenced by the desired signal interfering with itself during transmission.

3.6.6 NBI. Finally, a Narrow Band Interference (NBI) signal is introduced. Three signals based on an IEEE 802.11 [19] transmission are inserted over the range of -45 dB to 0 dB Signal-to-Interference Ratio (SIR) in increments of 2.0 dB, while maintaining a fixed SNR ($E_b/N_o = 7.5$ dB). This effectively creates a Signal-to-Interference-plus-Noise Ratio (SINR) ranging from -40 dB to -7.5 dB and -40 dB to -21.5 dB using TH-PPM and DS-BPSK, respectively. The first interferer uses DSSS-QPSK modulation in the $2.4 - 2.48$ GHz range. The second interferer is identical except that it operates at 5.0 GHz (middle of the UWB operating spectrum). The third interferer simulates a frequency hopping system ranging from 1.0 to 4.0 GHz, with hops occurring every microsecond.

3.6.7 Managing Experiments. Simulation of these various factors - SNR, code selection, transmitting users, multipath replications, modulation scheme, narrow band interferers, and synchronization amongst users - would take an excessive amount of time if all variables were tested individually. Thus, several factors are simulated simultaneously. Since 15 user levels, DS-BPSK and TH-PPM, and SNR are tested within each experiment, separate experiments need only occur for $[(5\ \textit{multipath replications} \times 6\ \textit{codes})] \times 2$ for the synchronous and asynchronous scenarios plus $3\ NB\ interferers \times 2\ \textit{codes}$ yielding a total of 66 experiments for length 31 codes.

Additional experimentation is done for $N_c = 127$ length Gold codes. It is expected that increased code length lowers BER at the expense of decreased data rate. Lastly, modulation scheme research is extended using 4-ary signalling with both PPM and BPSK modulations.

3.7 Evaluation Techniques and Validation

Simulation was chosen as the method of experimentation since it allows detailed control of parameters, large numbers of generated pulses, simple methods for altering generated codes, and graphical analysis and controlled determination of channel noise. Simulation is the most prudent choice given time and resource constraints.

Fundamentally, TH-PPM represents a form of orthogonal signaling and DS-BPSK represents a form of antipodal signaling. For communicating over an AWGN channel with a matched filter receiver, the theoretical probability of bit error P_b is

$$P_b = Q\left(\sqrt{\frac{E_b(1-\rho)}{N_o}}\right) \tag{3.3}$$

$$\begin{aligned} \rho &= 0 \qquad\qquad \text{(Orthogonal Signaling)} \\ \rho &= -1 \qquad\qquad \text{(Antipodal Signaling)} \end{aligned}$$

where Q is the complementary error function, E_b/N_o is the average energy per bit divided by the AWGN power spectral density, and ρ is the correlation coefficent [31].

The analytic expression (3.3) is used to validate model and simulation results for the single user case. To characterize MA communication performance for N_u total users, $(N_u - 1)$ users may be treated as noise (interference) from the perspective of a given user's receiver. Assuming each of the $(N_u - 1)$ interferers occupy a bandwidth consistent with the desired signal, an aggregate interference factor (I)

can be generated and incorporated into (3.3), resulting in the modified P_b expression

$$P_b = Q\left(\sqrt{\frac{E_b(1-\rho)}{N_o + I}}\right). \tag{3.4}$$

Equation 3.3 is analyzed to validate model results for the single user scenario using (1) the orthogonal representation for time hopping and (2) the antipodal representation for direct sequence multiple access. Once the single user model is validated, the model is applied to multiple users. Experimentation proceeds under the model's assumptions of AWGN with non-selective channel fading.

3.8 Workload

The SNR and total-number-of-users workloads submitted for these simulations is appropriate for the entire system under test. These factors affect operation throughout the system. The total number of pulses generated, the noise through the channel, and the ability of the receiver to detect and estimate data symbols are all directly impacted. Providing the simulation with 10 values for SNR and then 15 values for user levels, 6 values for code selection, 5 values for multipath replications, 3 NB interferers, and 2 scenarios for synchronization, is consistent with published literature and reasonable given the memory and CPU speed of the computers available.

3.9 Experimental Design

This research assesses the performance characteristics of a UWB communication system operating at a center frequency of 5.0 GHz. Though the trends reported should hold for any operating frequency, the parameters are fixed as described in Section 3.5 to indicate simulated performance in the unlicensed spectrum as authorized by the FCC for UWB systems.

Experimentation occurs in several phases. Initially, the benchmark of 300 bit errors is reduced to develop the code and gain a course understanding of the effect SNR has on the results. Once the code is fully developed, the model is validated. SNR is varied over 10 values, mapping to $E_b/N_o = 0$ to 10 dB, in increments of 1.0 dB. Upon validation of results with the expected theoretical results described in (3.3), experimentation proceeds to incorporate multiple users.

Initially, five code families are used to test multiple access interference, including Gold, Random Integer, SAC, SAA, and SAAC codes. These biphase codes are generated with length 31. The number of total users (including the transmitted signal of interest plus all interfering transmitters) is varied from 1 to 15 transmitters. The SNR is fixed at $E_b/N_o = 10$ dB. All trials are recorded for a synchronous network of users. These trials are repeated for users transmitting asynchronously.

MPI is added by creating $N_{MP} = 0, 2, 5, 10$ and 40 multipath reflections per user and recording the synchronous and asynchronous results for a fixed SNR.

Since the code generation process is statistical, the sixth code case is created from the original SAC code. The first five SAC codes are reordered to appear as the last five users out of the 15 total. The MPI and MAI tests are replicated for this code selection. This tests the significance of the ordering for the Simulated Annealing codes.

Additional experiments are conducted to test NBI and various length codes as outlined in Section 3.6.

Based on the central limit theorem and AWGN channel properties, replications of these trials should have a normal distribution. Experimental replications continue until the standard deviation falls within a 95% confidence interval according to

$$C.I. = \bar{x} \pm z\frac{s}{\sqrt{n}} \tag{3.5}$$

where $z = 1.96$, s is the standard deviation, and $\bar{x}$ is the mean of P_b [18].

To insure the model is working properly and the benchmark of 300 errors is sufficient for the desired confidence interval, two test cases are run in which the MA experiments for 15 users using Gold and RI codes at $N_{MP} = 0$ and 10 replications are repeated 3 times each, providing 3 independent P_b values for each user level for a given code and N_{MP}. The experimental error is calculated against the mean of the 3 trials for each case and a quantile-quantile plot of the residuals is produced, verifying the errors are normally distributed. The sum of the squared errors is

$$SSE = \sum_i \sum_j e_{ij}^2 \tag{3.6}$$

where the error in the j^{th} replication of the i^{th} experiment can be found from $e_{ij} = y_{ij} - \hat{y}_i$. The term y_{ij} is the value of P_b in this case for a particular replication of the experiment, and $\hat{y}_i$ is the average value of the i^{th} experiment.

The residual quantity $\sum_j e_{ij}^2$ is plotted against normal quantiles estimated by $x_i = 4.91[q_i^{0.14} - (1 - q_i^{0.14})]$ where $q_i = (i - 0.5)/n$ and n is the total number of experiments [18]. If the plot of residuals versus normal quantiles is linear, then the experimental error is normal and no bias exists in the simulations. A goodness of fit test known as the Shapiro-Wilk W Test is performed to quantify the normality of these errors [28].

3.10 Analyze and Interpret Results

Successful results must support the goals of this research to determine, for particular error levels, the SNR required and multiple access interference experienced when using spreading codes for time hopping and direct sequence UWB communications. P_b is plotted versus SNR for various multiple access codes. Fixing the value of SNR, P_b is plotted versus the number of users given the different codes. Results are anticipated to follow a logarithmic scale between 0.5 and 10^{-6} for P_b over the range of SNR values. Similar results are expected when P_b is plotted over the range of user levels. The values of P_b are distinquishable at various factor levels so that only visual tests are needed to determine uniqueness, avoiding the need for a t-test [18] to determine statistically unique values. An analysis of variance (ANOVA) [18] is performed to compare the relative influences of code type, user level, and multipath. Using methods similar to that for finding SSE, the sum squared values are determined. These values are divided by the degrees of freedom, based on the total number of samples, to find the mean square value of a given factor "x" (MSx). Since each of these values are normally distributed according to the central limit theorem, a ratio of MSx to the mean square of the error (MSE) yields an F-distribution. If this ratio is higher than unity, the factor is deemed a significant cause of the variation. From the ANOVA process, the significant factors affecting the output value P_b are determined.

The effects of the significant factors are also quantified by determining the average *BER Improvement* (over 15 users) for each code relative to a Gold code baseline. Since a ratio can be determined for the P_b value of one code relative to that of another code, a decibel value is used to report the BER improvement. The average ratio of P_b values is calculated for the 15 users. From this improvement factor, the performance of each code can be assessed. Analysis then proceeds to associate the improvement factor trend with the expected code performance based upon the code metrics Peak Sidelobe Levels (PSL), Integrated Sidelobe Levels (ISL), and Peak Cross-Correlation Levels (PCCL) which measure the auto- and cross-correlation properties of the codes.

3.11 Summary of Experimental Setup

This chapter outlines the methodology used to assess the performance of a multiple access UWB communication system using TH-PPM and DS-BPSK modulations. The entire communication system is characterized with specific emphasis on the multiple access and modulation components. Since the system's service is transmitting data bits, the metrics used to characterize system performance are probability of bit error versus E_b/N_o and user capacity.

The research is conducted using simulation in two phases. Initial results are used to validate the simulation model and verify values to be used in subsequent simulations. The second phase conducts the experiments.

Analysis of the results compares the relative effects of SNR, generated code, user level, number of multipaths, narrow band interference, and synchronization. Results are expected to give an indication of how these factors affect time hopping and direct sequence spreading of UWB communications.

IV. Results and Analysis

4.1 Code Metric Analysis

Six codes were used for analysis, including Gold, Random Integer (RI), and Simulated Annealing (SA) with generated emphasis on (1) equally weighting PSL and PCCL (SAAC), (2) minimizing PSL (SAA), (3) minimizing PCCL (SAC), and (4) a reordered set of the SAC code (SACmod).

4.1.1 TH-PPM Metrics. Table 4.1 shows the number of times (# Collisions) users assign the same time slot over one full TH-PPM UWB code period when $N_u = 2$ through 15 users are present in a synchronous network. The maximum number of possible collisions over one code period equals $(N_c = 31) \times [(N_u - 1) = 14] = 434$. For 15 users, collisions occur on average between every second-to-fourth time slot for RI and the various SA codes. As indicated, the Gold code set yielded zero collisions for all cases.

Table 4.1 31-Bit TH-PPM Code Characteristics: Single Code Collisions/User

	Number of Users														Avg./
Code Family	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Slot
Gold	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RI	5	11	11	13	13	14	14	14	15	16	17	17	18	18	0.58
SAAC	0	0	0	0	0	0	0	0	2	3	3	3	3	8	0.26
SAA	0	0	0	7	7	7	9	11	11	11	11	13	15	17	0.55
SAC	1	1	1	1	1	2	2	6	6	9	9	9	9	9	0.29
SACmod	7	7	7	7	7	7	7	7	9	9	10	14	14	17	0.55

The number of collisions in a time slot can also be viewed from a network perspective, disregarding which of the 15 codes is designated as the desired signal. In other words, rather than recording only the number of collisions occurring between the desired signal and the other 14 coded transmitting signals in a given time slot, each slot is viewed from a code family perspective and the total number of codes

Table 4.2 31-Bit TH-PPM Code Characteristics: Code Family Perspective

Code Family	# Collisions	Avg. # Collisions/Slot
Gold	0	0
RI	108	3.48
SAAC	106	3.42
SAA	113	3.65
SAC	105	3.39
SACmod	105	3.39

colliding in a given time slot is recorded during one code period of the time hopped signal. This yields a maximum number of possible collisions over one code period equal to $(N_c = 31) \times [(N_u - 1) + (N_u - 2) + ... + 1] = 3255$. From a network perspective, an average of three to four collisions occurred in each time slot for all codes except Gold codes, which again reported no collisions in a synchronous network of 15 users.

A graphical representation of the number of collisions occurring within each of the time slots for length $N_c = 31$ code periods is presented in Appendix A for each of the six codes tested.

4.1.2 DS-BPSK Metrics. Worst case (maximum peak) periodic correlation characteristics for the six code families, each containing a collection of 15 codes, are given in Table 4.3. The maximum PSL and ISL values are taken from correlations across the entire code family. The maximum PCCL is shown for each user level to allow incremental code comparison.

These code metrics are calculated from the perspective of the individual, desired signal. The PSL, ISL, and PCCL are calculated by correlating the desired signal with the other users (codes) of a given family. Since simulations are performed by designating one coded signal from a code family as the desired signal for detection and estimation by the receiver, and the other 14 codes are therefore interfering signals, Table 4.3 provides code metrics that are consistent with the simulated results.

Table 4.3 31-Bit DS-BPSK Code Characteristics: Single Code Metrics/User

			PCCL(dB) for Given Number of Users					
Code Family	PSL(dB)	ISL(dB)	2	3-6	7	8	9	10-15
Gold	-10.74	0.53	-10.7	-10.7	-10.7	-10.7	-10.7	-10.7
RI	-10.74	-1.89	-6.31	-6.31	-6.31	-5.22	-5.22	-5.22
SAAC	-12.93	-3.18	-7.55	-7.55	-5.22	-5.22	-5.22	-5.22
SAA	- 9.00	-0.99	-9.00	-6.31	-6.31	-6.31	-6.31	-6.31
SAC	-10.74	-1.36	-7.55	-7.55	-7.55	-7.55	-6.31	-4.25
SACmod	- 5.22	3.97	-6.31	-6.31	-6.31	-6.31	-6.31	-6.31

Again, it is also useful to view code family metrics to gain insight into the effectiveness of a given code for operation from a network perspective, rather than a single user perspective. Table 4.4 provides numerical analysis of code metrics, specifically describing the maximum peak correlations over all combinations of 15 codes for a given code family.

Table 4.4 31-Bit DS-BPSK Code Characteristics: Code Family Perspective

Code Family	PSL(dB)	ISL(dB)	PCCL(dB)
Gold	-10.74	1.35	-10.74
RI	- 5.22	2.66	- 4.25
SAAC	- 9.00	0.47	- 5.22
SAA	- 9.00	-0.55	- 4.25
SAC	- 5.22	3.97	- 4.25
SACmod	- 5.22	3.97	- 4.25

4.2 Error Analysis

Error analysis was performed to determine model accuracy for simulations developed for this work. Quantile-quantile plots were generated using three replicated experiments with the arbitrarily chosen Gold, RI, and SAC codes to determine residuals for the $N_{MP} = 0$ and $N_{MP} = 10$ cases (cf., Section 3.9). The following quantile-quantile plots demonstrate approximate linearity of the residuals.

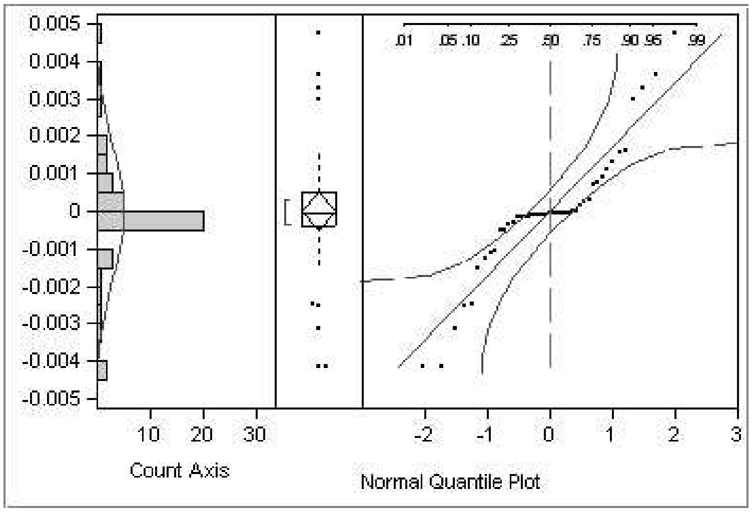


Figure 4.1 TH-PPM Quantile-Quantile Plot for P_b, $N_{MP} = 0$

The quantile plots reveal that nearly all residual data falls within the 95% confidence interval, which is drawn as black, hyperbolic arcs surrounding the linear fit. An exception to this is the residual plot of DS-BPSK modulation with $N_{MP} = 0$ shown in Fig. 4.2 where the quantile-quantile plot is nonlinear. This is due to the large variation in performance of the three codes tested under DS-BPSK and no MPI conditions. Figures 4.7, 4.8, and 4.9 provide quantile-quantile plots of the errors in P_b for Gold, RI, and SAC codes, respectively. Isolating the code factor reveals that the P_b residuals are normal (linear) and all results are reported within the 95% confidence interval. Table 4.5 indicates that the Shapiro-Wilk W Test also justifies the normality assumption of the errors. For TH-PPM modulation, the W statistic reported above 90% and the probability that the actual value falls below this level is 10% for the closest fit and less than 0.1% for the remaining case, indicating the normal model is a "good" fit. The lowest W statistic reported (69%) is also explained by the significant variation in code performance in DS-BPSK modulation. Isolation of the codes reveals W statistics of 97%, 85%, and 90%.

The residual plots of Figs. 4.5 and 4.6 indicate a uniform distribution of errors centered around a zero mean. As the probability of bit error increases, the

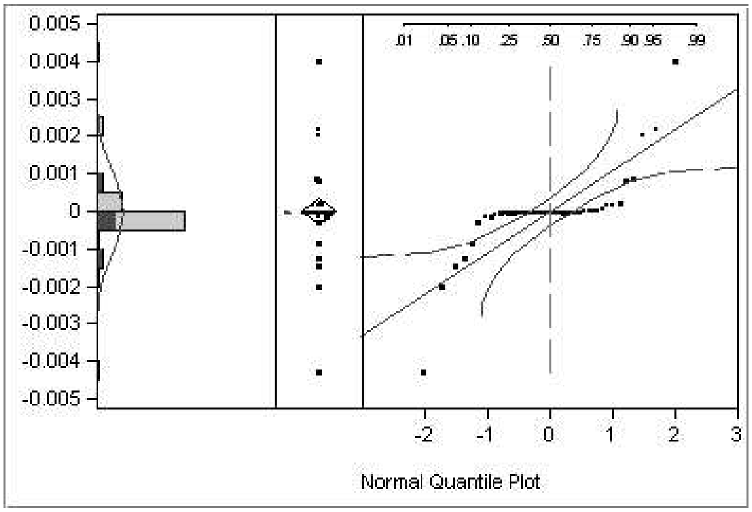


Figure 4.2 DS-BPSK Quantile-Quantile Plot for P_b, $N_{MP} = 0$

variance of the distribution of errors around the mean increases (as expected) since the experiment was designed to determine P_b values within ±11% accuracy of the actual value as described in Section 3.4. Therefore, as the value of P_b increases, the absolute value of the error is allowed to increase so that in the worst case of $P_b = 0.5$, ±11% error allows for an inaccurate reporting of P_b by as much as 0.055 as shown in Figs. 4.5 and 4.6.

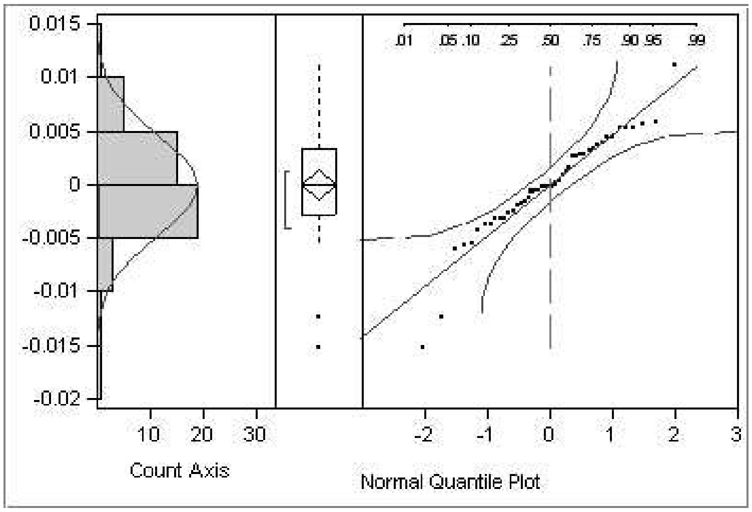


Figure 4.3 TH-PPM Quantile-Quantile Plot for P_b, $N_{MP} = 10$

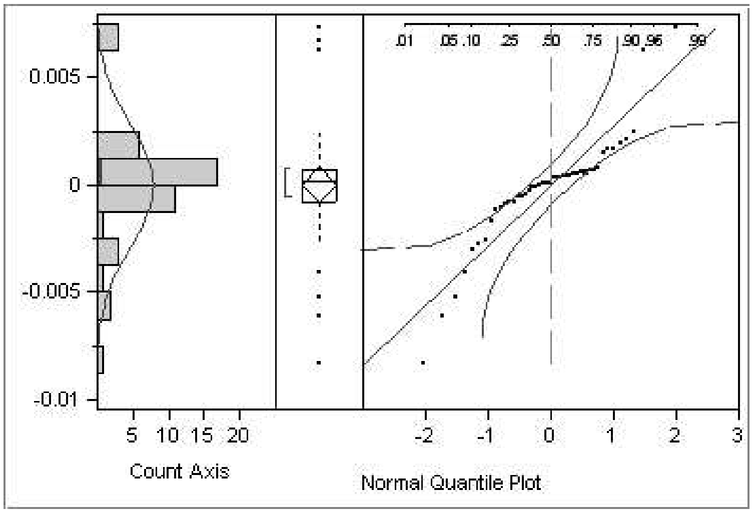


Figure 4.4 DS-BPSK Quantile-Quantile Plot for P_b, $N_{MP} = 10$

Table 4.5 Shapiro-Wilk W Test for Gold & RI Codes

MA/Modulation	N_{MP}	$W(\%)$	$Prob < W$
TH-PPM	0	90.17	0.0008
DS-BPSK	0	69.20	0.0001
TH-PPM	10	95.31	0.1044
DS-BPSK	10	89.60	0.0005

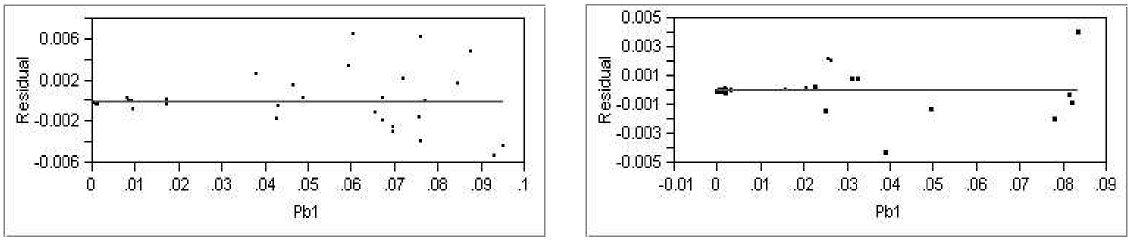


Figure 4.5 Residual Plots $N_{MP} = 0$: TH (Left) and DS (Right)

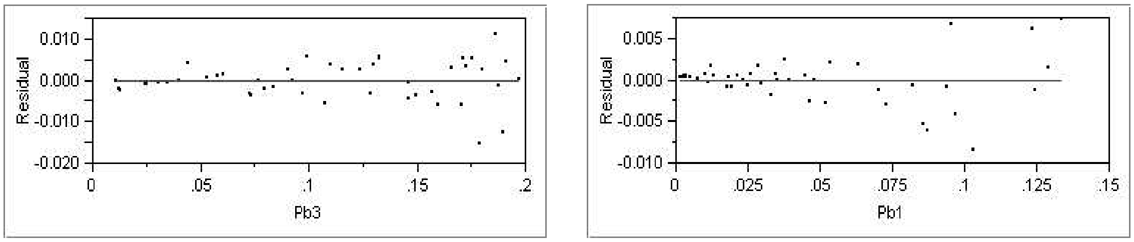


Figure 4.6 Residual Plots $N_{MP} = 10$: TH (Left) and DS (Right)

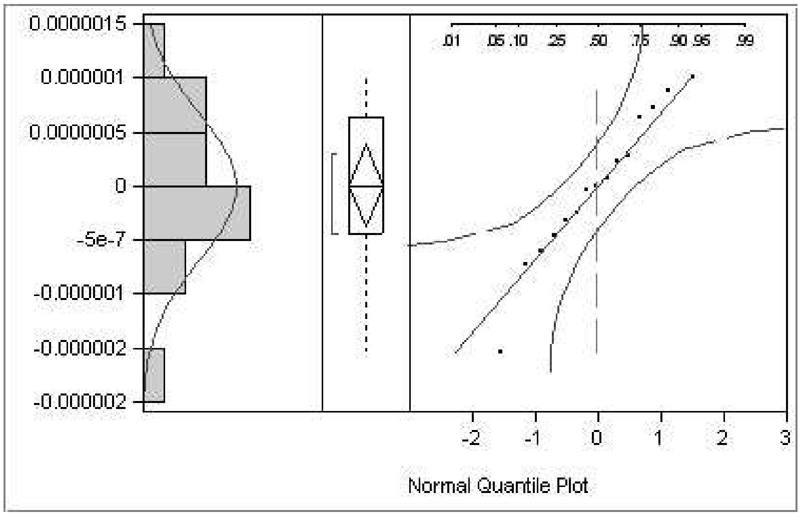


Figure 4.7 Gold DS-BPSK Quantile-Quantile Plot for P_b, $N_{MP} = 0$

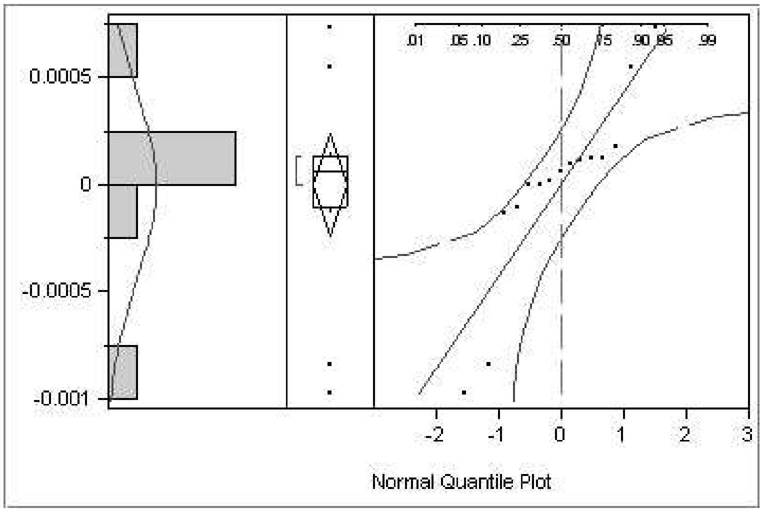


Figure 4.8 RI DS-BPSK Quantile-Quantile Plot for P_b, $N_{MP} = 0$

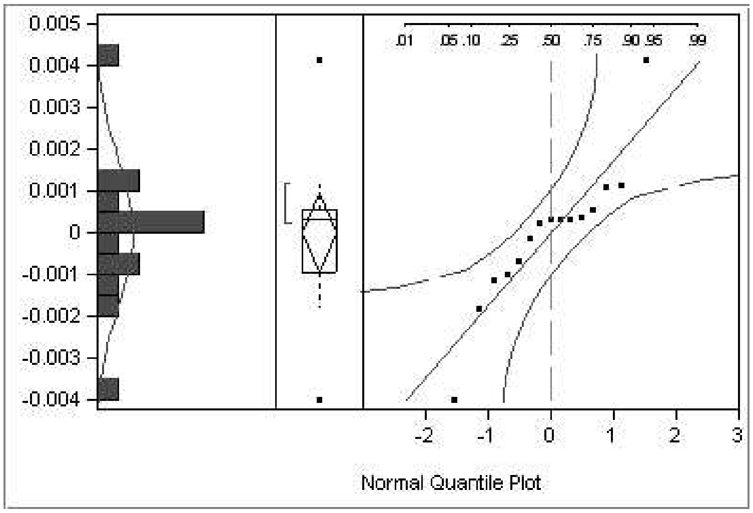


Figure 4.9 SAC DS-BPSK Quantile-Quantile Plot for P_b, $N_{MP} = 0$

4.3 ANOVA

An analysis of variance (ANOVA) was performed to determine the statistical significance of the simulation factors. A factorial analysis for both synchronous and asynchronous networks was performed. All six code types, five levels of multipath interference ($N_{MP} = 0, 2, 5, 10$ and 40), and all 15 user levels were used, accounting for 450 P_b values for the TH-PPM and DS-BPSK cases.

Table 4.6 lists the total percentage of P_b variation accounted for by each factor in a synchronous network. Since these experiments were controlled simulations, all variance is accounted for by these factors. It is observed that the interaction of factors *Code*Users* and *Users**N_{MP} accounted for the greatest percentage of variation in both modulation techniques.

Table 4.6 P_b Variation: Synchronous TH & DS

	Percentage of Variance	
Factor(s)	TH-PPM	DS-BPSK
Code	0.42%	0.08%
Users	0.96%	0.54%
N_{MP}	5.41%	2.00%
Code*Users	15.62%	39.28%
Code*N_{MP}	7.79%	11.69%
Users*N_{MP}	62.80%	37.22%
Code*Users*N_{MP}	7.00%	9.19%

For synchronous transmitters, almost all factors reported F ratios much greater than unity, indicating significant impact on the communication performance of the simulated UWB system [18]. The exception was in DS for the *Code* factor where the F ratio equals 0.4625. However, it is misleading to regard this factor as insignificant. All MPI levels are used, meaning an insensitivity to code selection at higher N_{MP} levels conceals the impact of code type when little or no MPI exists. The probability that the F ratio is actually higher than the reported value is 80.4%, another indicator that code type alone cannot necessarily be ignored. When $N_{MP} = 0$, the ANOVA results in Table 4.7 indicate *Code* is a significant factor.

Table 4.7 P_b Variation With No MPI: Synchronous TH & DS

Source	Percentage of Variance	
Factor(s)	TH-PPM	DS-BPSK
Code	53.26%	29.80%
Users	29.04%	47.23%
Code*Users	17.70%	22.98%

Table 4.8 lists the total percentage of P_b variation accounted for by each factor in an asynchronous network. It is observed that the interaction of factors *Users**N_{MP} accounted for the greatest percentage of variation in both modulation techniques.

Table 4.8 P_b Variation: Asynchronous TH & DS

Source	Percentage of Variance	
Factor(s)	TH-PPM	DS-BPSK
Code	0.008%	0.55%
Users	0.32%	0.16%
N_{MP}	2.89%	0.15%
Code*Users	1.20%	4.46%
Code*N_{MP}	0.57%	6.20%
Users*N_{MP}	91.29%	77.51%
Code*Users*N_{MP}	3.73%	10.97%

For asynchronous transmitters, all factors had F ratios greater than unity, with three exceptions. In TH, *Code* reported an F ratio of 0.1228 with a 98.7% chance of actually having a higher value. The same caution of ignoring factor significance is expressed here as in the synchronous case. The other two exceptions were for DS, where the F ratios for *Users* and N_{MP} were 0.2836 (Prob > F = 0.9953) and 0.9806 (Prob > F = 0.4184), respectively. ANOVA results for an asynchronous network when $N_{MP} = 0$ are presented in Table 4.9. In TH-PPM modulation when no MPI is present, *Code* accounts for 34.93% of the variation in P_b. *Code* accounts for less than 1% of the variation in an asynchronous DS-BPSK modulated network, indicating it is not a significant factor.

Table 4.9 P_b Variation With No MPI: Asynchronous TH & DS

Source	Percentage of Variance	
Factor(s)	TH-PPM	DS-BPSK
Code	34.93%	0.42 %
Users	52.75%	97.68%
Code*Users	12.32%	1.90 %

Additional information for quantifying the extent to which BER is affected for each factor level and the relationship of performance to defined code metrics is presented in the remaining sections of this chapter.

4.4 Communication Performance

For validation with previous UWB research results, simulations were run for all models in the absence of Multiple Access (MA) coding and Multipath (MP) interference. Following pulse generation, data modulation, signal transmission, and reception, communication symbols were demodulated and the total number of bit errors recorded.

Communication performance results are shown in Fig. 4.10 [4]. As indicated, simulation results are consistent with analytic results of (3.3) for both orthogonal TH-PPM (solid line) and antipodal DS-BPSK (dashed line) modulations. The solid (TH-PPM) and dashed (DS-BPSK) line convention is maintained for presentation of research results. Simulated communication performance for the TH and DS UWB techniques approximate theoretical orthogonal and antipodal signaling P_b, reflecting a mean absolute error of 1.6×10^{-6} and standard deviation of 5.7×10^{-4}.

4.5 Synchronous Network

4.5.1 MAI Results for $N_c = 31$ Length Codes. Multiple access interference (MAI) effects are characterized via simulation for UWB TH-PPM and DS-BPSK modulations using up to $N_u = 15$ users in a *synchronous network*. For all simulation results, the received power in *all* undesired interfering signals is identical to the

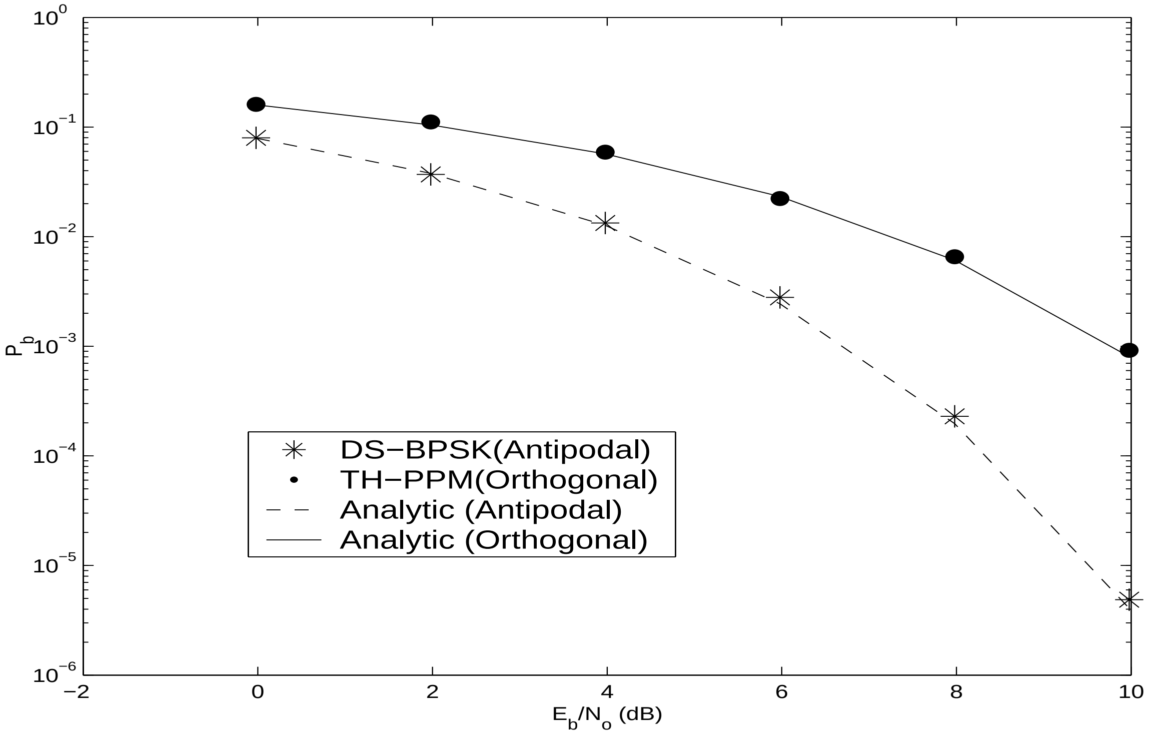


Figure 4.10 Communication Performance for AWGN Channel [4]

received power of the desired signal. Previously published results for UWB systems using Gold code sequences provided a baseline for validating the model [11], [32].

Using a fixed $E_b/N_o = 10$ dB with no MPI present, MAI simulations were run for both TH-PPM and DS-BPSK modulations with results presented in Figs. 4.11 and 4.12, respectively. Under these conditions for both modulations, the Gold coded network maintained a lower BER for each MAI level; P_b varied by 50% for TH and one order-of-magnitude for DS from 1 to 15 users. The RI and SA codes yielded approximately 2 and 4 orders-of-magnitude increase in P_b for TH-PPM and DS-BPSK, respectively.

In the synchronous TH-PPM case where $b = 5$ binary code values are mapped to one of $2^b = 32$ decimal-valued time slots, no time slot collisions occur for the $N_u = 15$ Gold code case. On average, collisions occur every second or third time slot for RI and SA codes. Data in Table 4.1 reflects the trend seen here for P_b. The

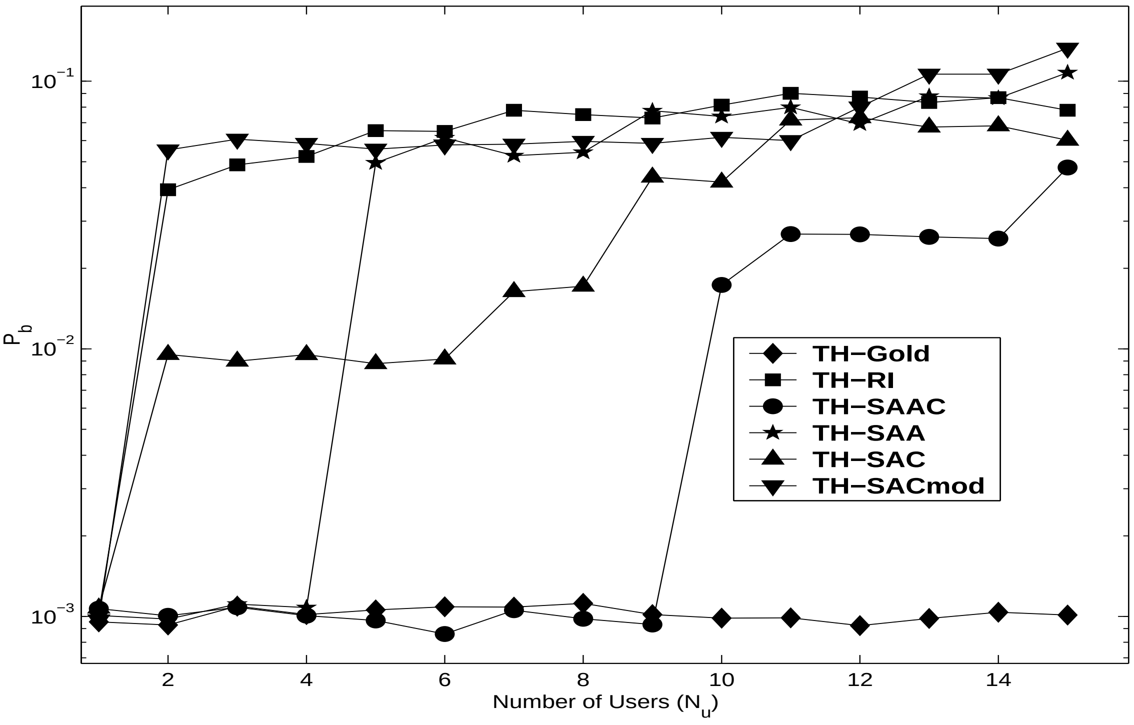


Figure 4.11 TH-PPM MAI Results, No MPI, Synchronous Network

increase in collisions is proportional to the BER increase as the number of users increases. Similarly, in the DS case, the significant robustness in BER performance afforded by Gold coding can be explained using data in Table 4.3. As indicated, the maximum PCCL for Gold codes is between 2 and 4 times (4.0 to 6.0 dB) lower than the other codes.

4.5.2 Single User Multipath Effects. Multipath interference (MPI) effects were characterized using an RMS time delay value of 15.4 *ns* for each user's replicated signal [11]. Data was generated for two scenarios operating at an $E_b/N_o = 10$ dB for various levels of multipath replication including $N_{MP} = 0, 2, 5, 10$ and 40 multipath replications per user. This scenario includes a network containing a single transmitter/receiver link ($N_u = 1$).

Multipath results for the single transmitter/receiver scenario are provided in Fig. 4.13 for TH-PPM modulation and Fig. 4.14 for DS-BPSK modulation for

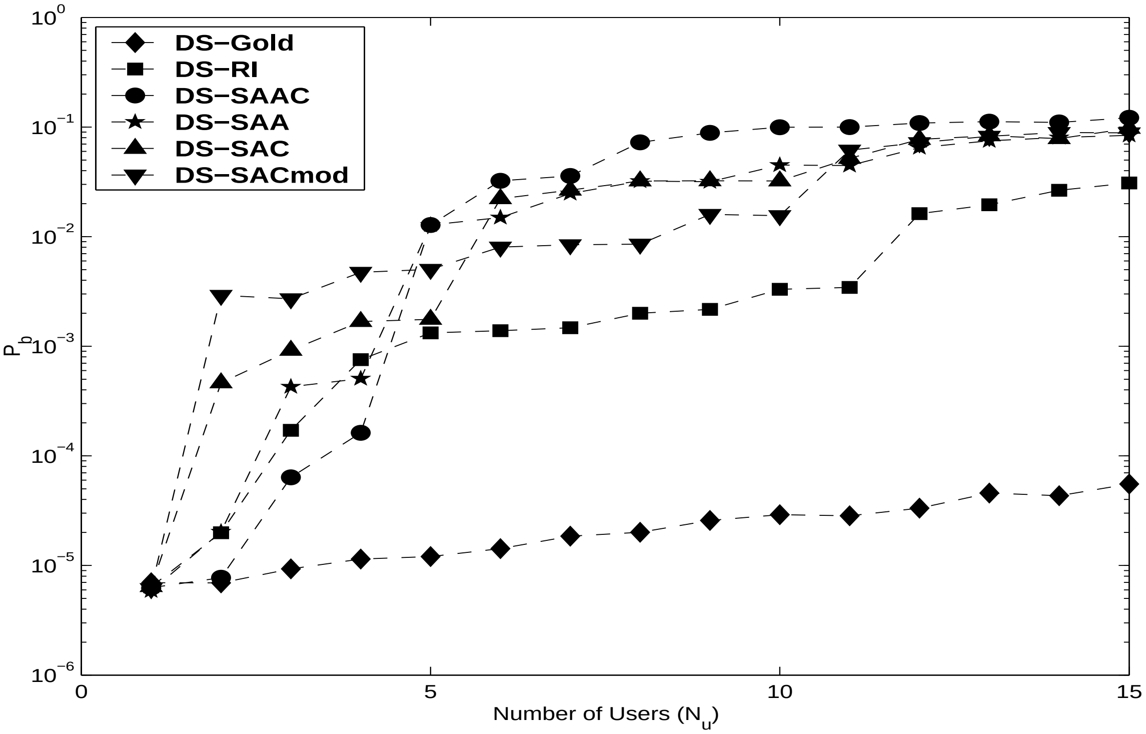


Figure 4.12 DS-BPSK MAI Results, No MPI, Synchronous Network

each code considered. In these instances, the receiver under test receives $[N_u = 1] \times [N_{MP} + 1]$ total signals, including one direct *desired* signal and N_{MP} delayed multipath *interfering* signals. Gold, RI, and all SA codes perform similarly under the simulated multipath conditions for each modulation. With the exception of code SACmod, there appears to be no appreciable multipath performance advantage for any code considered. The variation of SACmod performance in the DS case is explained by the PSL and ISL peaks since the interference signal is a delayed replica of the desired signal. For both metrics, the maximum peak for SACmod is at least twice as great as any of the other codes. The variation in performance by SACmod indicates the significance of code order (and which code is the desired signal) for SA coding.

The ratio of P_b levels at each value of N_{MP} to the baseline performance of P_b at $N_{MP} = 0$ is the average P_b loss factor for each modulation scheme. TH-PPM

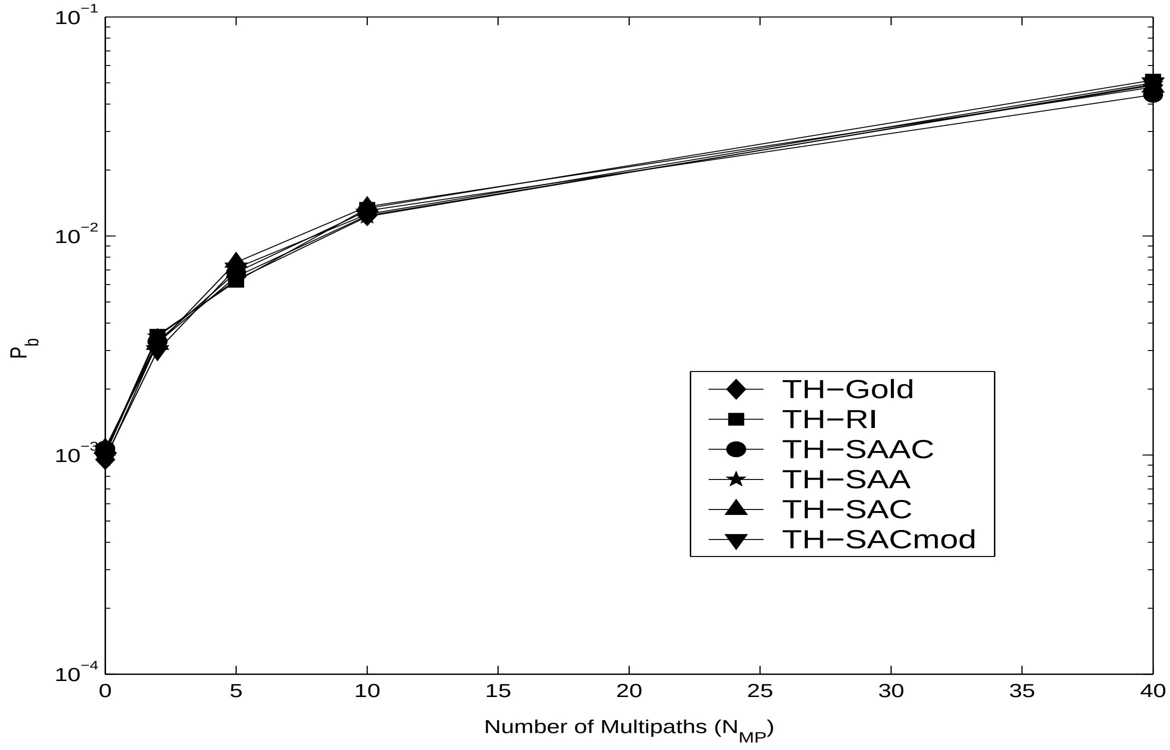


Figure 4.13 TH-PPM MPI Results: Single Transmitter/Receiver Link

averaged a factor of 1.5 loss in P_b per multipath replication, or P_b (degraded)=$[1.5 \times N_{MP} + 1] \times (P_b \; at \; N_{MP} = 0)$ over a P_b range of $[1 \times 10^{-3} : 5 \times 10^{-2}]$ for $N_{MP} = [0 : 40]$ replications. Likewise, over the same multipath range, average DS-BPSK performance is given by P_b (degraded)=$[25 \times N_{MP} + 1] \times (P_b \; at \; N_{MP} = 0)$ over a P_b range of $[8 \times 10^{-6} : 1 \times 10^{-2}]$. Graphical analysis of P_b versus multipaths indicates a linear relationship.

4.5.3 MAI and MPI Results. Multipath interference results were generated using $N_u = 1, 2, \ldots, 15$ synchronous users and $N_{MP} = 0, 2, 5, 10$, and 40 multipath replications. Representative results for $N_{MP} = 5$ are provided in Figs. 4.15 and 4.16 for TH-PPM and DS-BPSK modulations, respectively. In these cases, the receiver under test receives $N_u \times (N_{MP} + 1)$ total signals, including one direct *desired* signal, $(N_u - 1)$ direct MAI *interfering* signals and $(N_u) \times (N_{MP})$ delayed multipath *interfering* signals.

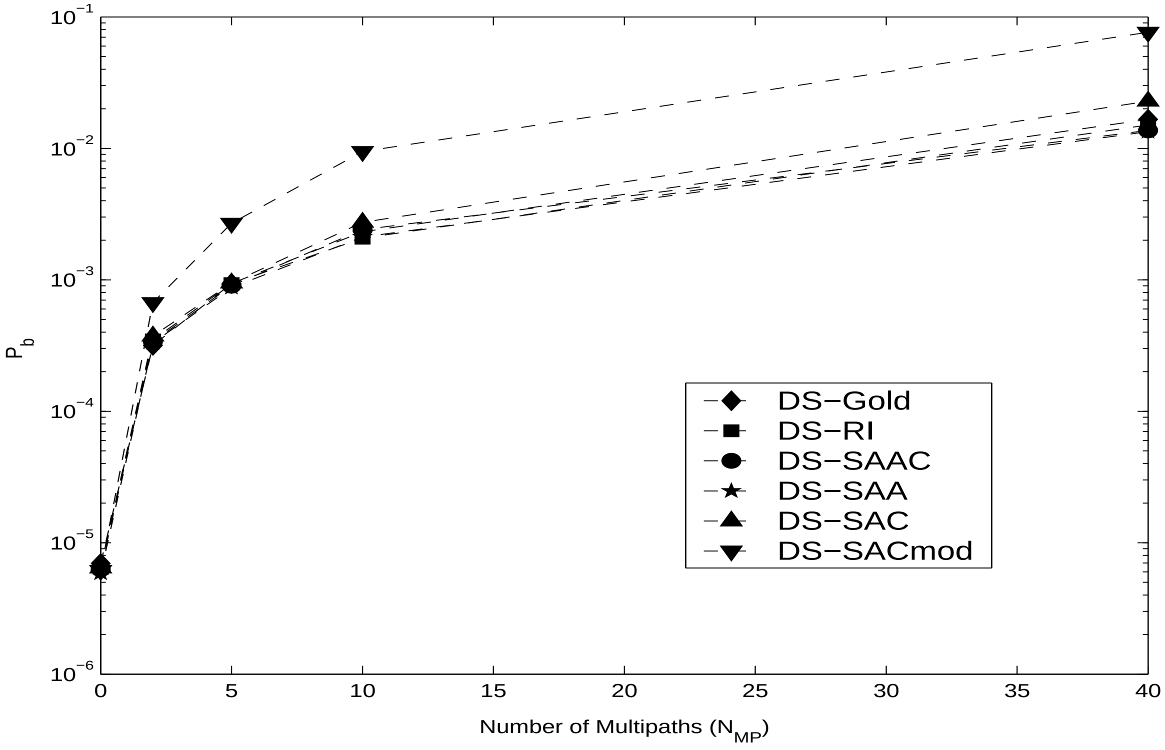


Figure 4.14 DS-BPSK MPI Results: Single Transmitter/Receiver Link

Results indicate all coding and modulation combinations exhibit expected P_b performance degradation as N_u increases. The variance in P_b between codes decreases as MPI increases, with code performance exhibiting the same trend established under MAI conditions without MPI. Using the Gold code and DS-BPSK modulation has the same degrading trend as N_u increases but appears more robust in terms of absolute degradation. Similar results were obtained for the $N_{MP} = 2, 10$, and 40 cases and are provided in Appendix A.

Tables 4.10 and 4.11 show average *BER Improvement*. This improvement is the average ratio (over 15 users) of P_b for a given code to that of the Gold code (designated as baseline performance). The more negative the decibel number reported, the greater the improvement yielded by using Gold codes versus the given code for the particular modulation and multipath level. For example, in a synchronized network containing up to 15 users, TH-PPM modulation with Gold coding provides an

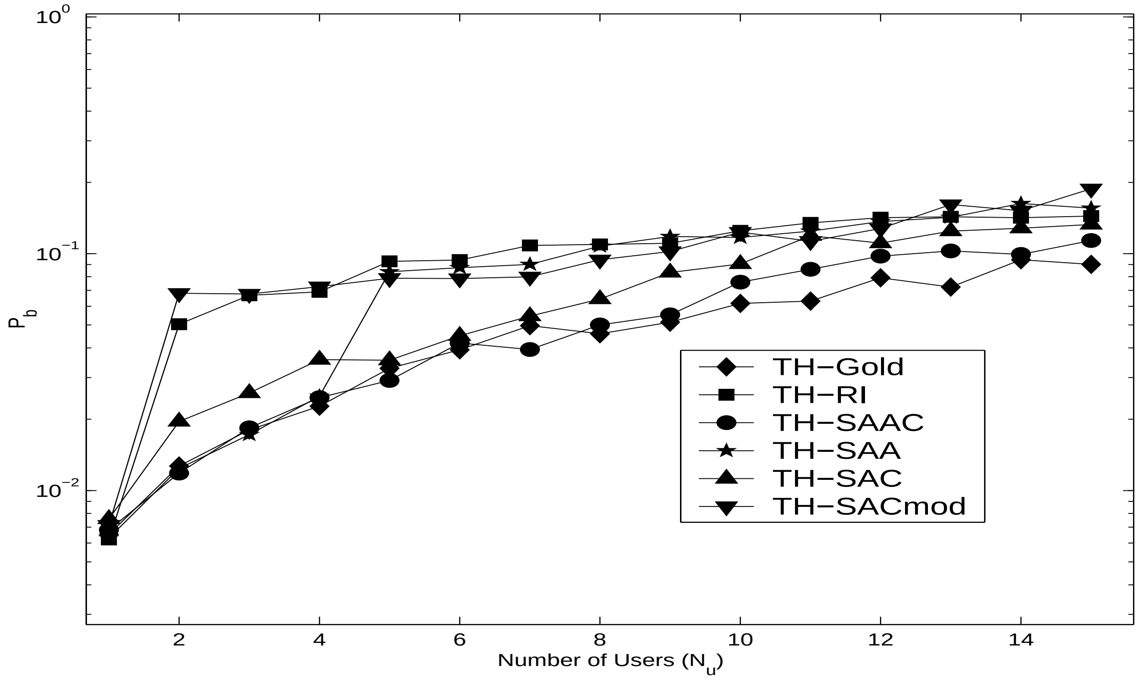


Figure 4.15 TH-PPM MAI Results, $N_{MP} = 5$ Replications, Synchronous Network

average P_b improvement of -17.1 dB and -13.0 dB relative to RI and SAC codes, respectively. Likewise, DS-BPSK modulation with Gold coding provides an average P_b improvement of -18.9 dB and -26.8 dB relative to RI and SAC codes, respectively.

Table 4.10 Average TH-PPM BER Improvement(dB): Synchronous Network

Code	Number of Multipath Replications (N_{MP})				
Family	0	2	5	10	40
RI	-17.1	-6.21	-3.38	-2.12	-0.35
SAAC	-5.64	-1.08	-0.38	-0.25	-0.04
SAA	-13.5	-4.26	-2.21	-1.13	-0.10
SAC	-13.0	-3.20	-1.48	-0.81	-0.20
SACmod	-17.0	-6.01	-3.32	-2.00	-0.41

In TH-PPM modulation, the number of collisions (cf., Tables 4.1 and 4.2) exactly characterizes the relative P_b performance of various codes. In DS-BPSK modulation, the P_b performance of the various codes loosely follows PSL, ISL, and

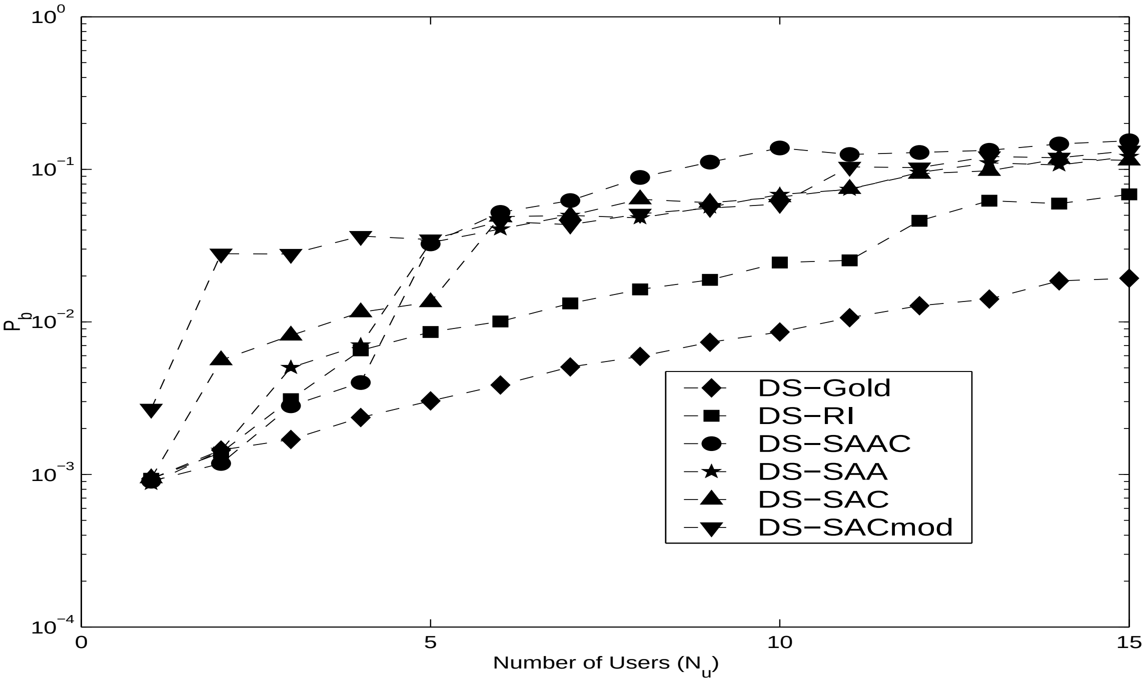


Figure 4.16 DS-BPSK MAI Results, $N_{MP} = 5$ Replications, Synchronous Network

PCCL trends of Tables 4.3 and 4.4. Gold codes possess the lowest maximum PCCL and achieve the lowest BER.

Table 4.11 Average DS-BPSK BER Improvement(dB): Synchronous Network

Code	Number of Multipath Replications (N_{MP})				
Family	0	2	5	10	40
RI	-18.9	-7.07	-3.94	-1.84	-0.09
SAAC	-26.3	-12.6	-8.00	-4.56	-0.61
SAA	-25.6	-11.5	-7.22	-3.88	-0.58
SAC	-26.8	-12.2	-7.77	-4.65	-0.99
SACmod	-26.8	-13.5	-9.56	-6.37	-2.52

4.6 Asynchronous Network

Asynchronous network characterization uses the same procedure as reported above for the synchronous network except that users transmit using randomly selected offsets within (1) chip interval T_c for TH-PPM modulation and (2) PRI T_o for DS-BPSK modulations. This section provides asynchronous network results using a presentation format consistent with the synchronous results.

4.6.1 MAI Results for $N_c = 31$ Length Codes. Multiple access interference (MAI) effects are characterized via simulation for UWB TH-PPM and DS-BPSK modulations using up to $N_u = 15$ users in an *asynchronous network*. For all simulation results, the received power in *all* undesired interfering signals is identical to the received power of the desired signal.

Using a fixed $E_b/N_o = 10$ dB with no MPI present, MAI simulations were run for both TH-PPM and DS-BPSK modulations with results presented in Figs. 4.17 and 4.18, respectively. Under these conditions with TH-PPM modulation, the SAAC coded network maintained a lower BER; P_b varied by approximately one order-of-magnitude for all codes. In the DS-BPSK modulation case, the SAC coded network maintained a lower BER with P_b for all codes varying by 2.5 orders-of-magnitude from 1 to 15 users. The variance for all codes was lower over the range of 15 users in the asynchronous case than in the synchronous network where the P_b change spanned several orders-of-magnitude. It is interesting to note that despite the increase in Gold code BER for asynchronous communications (thus, the overall best performance has decreased relative to a synchronized network), the highest BERs reported for both modulations over an asynchronous network with no MPI were 2×10^{-2} (TH-PPM) and 2×10^{-3} (DS-BPSK). For the synchronous network, the poorest performing codes reached much higher error levels, approximately 1×10^{-1} for both TH-PPM and DS-BPSK modulations.

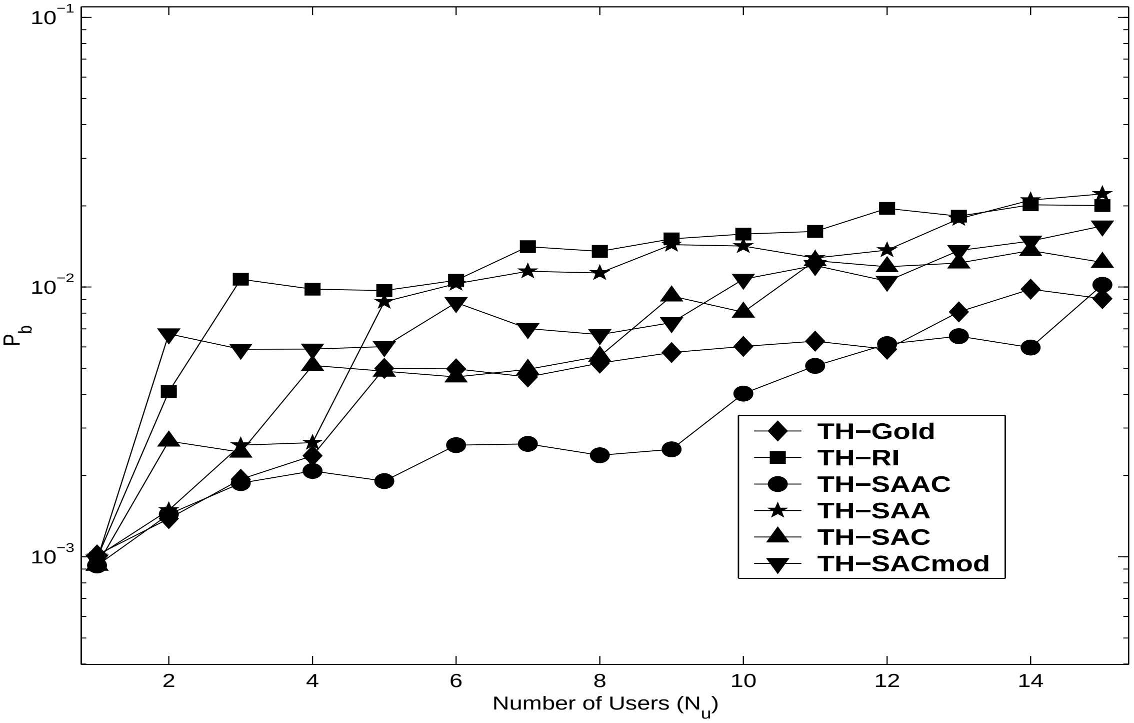


Figure 4.17 TH-PPM MAI Results, No MPI, Asynchronous Network

4.6.2 MAI and MPI Results. Multipath interference results were generated using $N_u = 1, 2, \ldots, 15$ asynchronous users and $N_{MP} = 0, 2, 5, 10$, and 40 multipath replications. Representative results for $N_{MP} = 5$ are provided in Figs. 4.19 and 4.20 for TH-PPM and DS-BPSK modulations, respectively. In these cases, the receiver under test receives $N_u \times (N_{MP} + 1)$ total signals, including one direct *desired* signal, $(N_u - 1)$ direct MAI *interfering* signals and $(N_u) \times (N_{MP})$ delayed multipath *interfering* signals.

As indicated by the results, all coding and modulation combinations exhibit expected P_b performance degradation as N_u increases. For TH-PPM modulation, the variance in P_b between codes decreases as MPI increases, with code performance exhibiting the same trend established under MAI conditions without MPI. In the DS-BPSK modulation case, trends are similar except that SAC codes performed best with MAI and no MPI. When MPI is present, SAC code performance degrades so

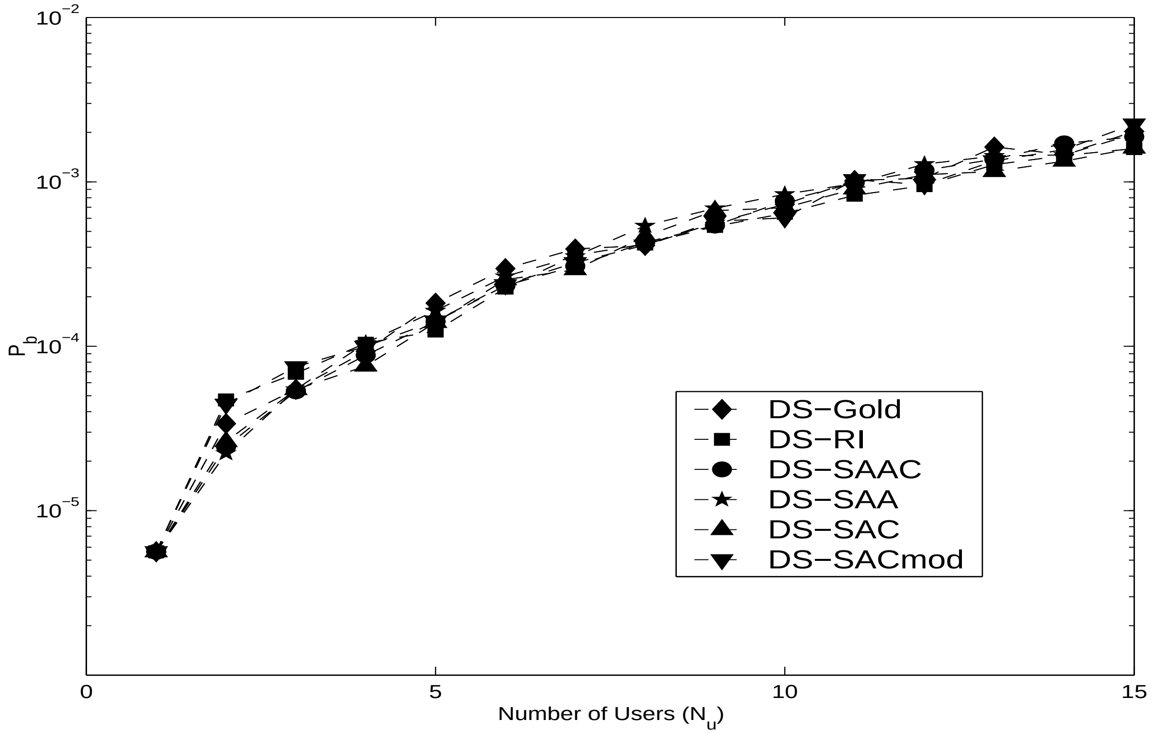


Figure 4.18 DS-BPSK MAI Results, No MPI, Asynchronous Network

that SAAC codes perform slightly better on average. Gold coding yields the best average performance out of the six codes tested for N_{MP} =2, 5, and 10 multipath replications. At $N_{MP} = 40$ replications, all SA code configurations average slightly better BER performance than Gold codes using TH-PPM modulation. For DS-BPSK modulation, SAAC codes average lower BER than the other codes tested in an asynchronous network. Simulation results for $N_{MP} = 2, 10$, and 40 replications are provided in Appendix A.

Tables 4.12 and 4.13 show average *BER Improvement* for an asynchronous network containing up to $N_u = 15$ users. Without MPI, TH-PPM modulation with Gold coding exhibits an average improvement of -4.07 dB relative to RI codes. SAAC coding provides lower BER (approximately 1.5 dB) than Gold coding. Likewise, DS-BPSK modulation with RI (0.30 dB) and SAAC (0.48 dB) coding provides lower BER than Gold coding.

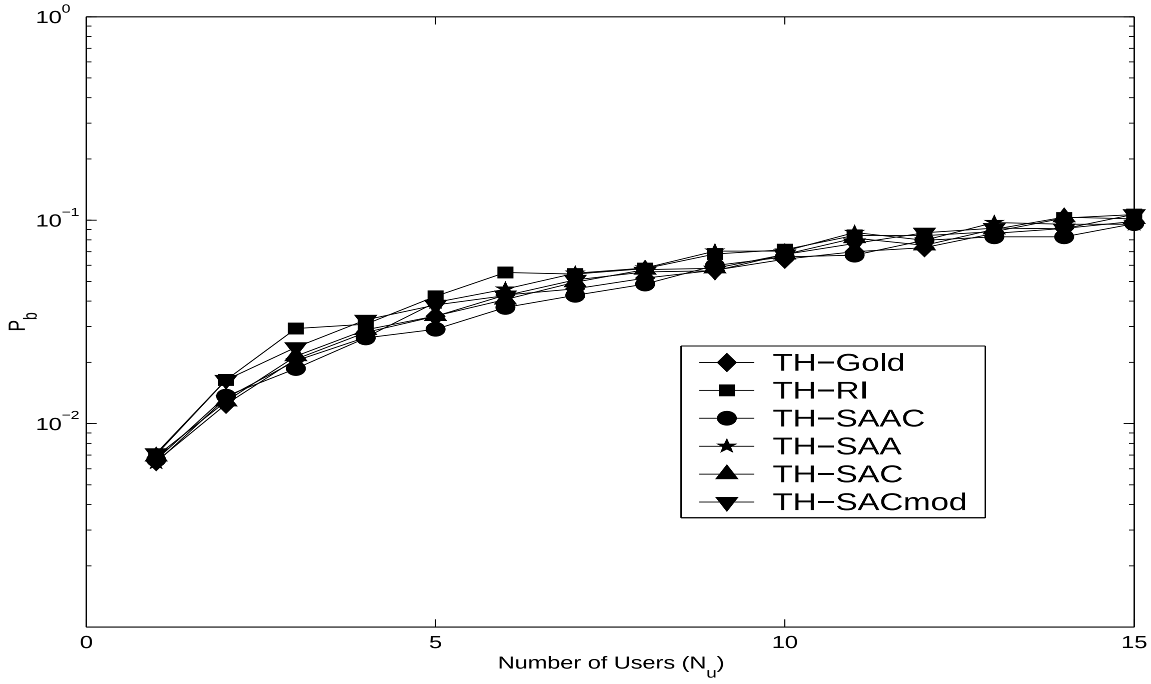


Figure 4.19 TH-PPM MAI Results, $N_{MP} = 5$ Replications, Asynchronous Network

The demonstrated change in relative performance (synchronous versus asynchronous) of the different codes is reasonable since the benefits of creating codes with fewer collisions per time slot and lower correlation sidelobe peaks are diminished in asynchronous networks.

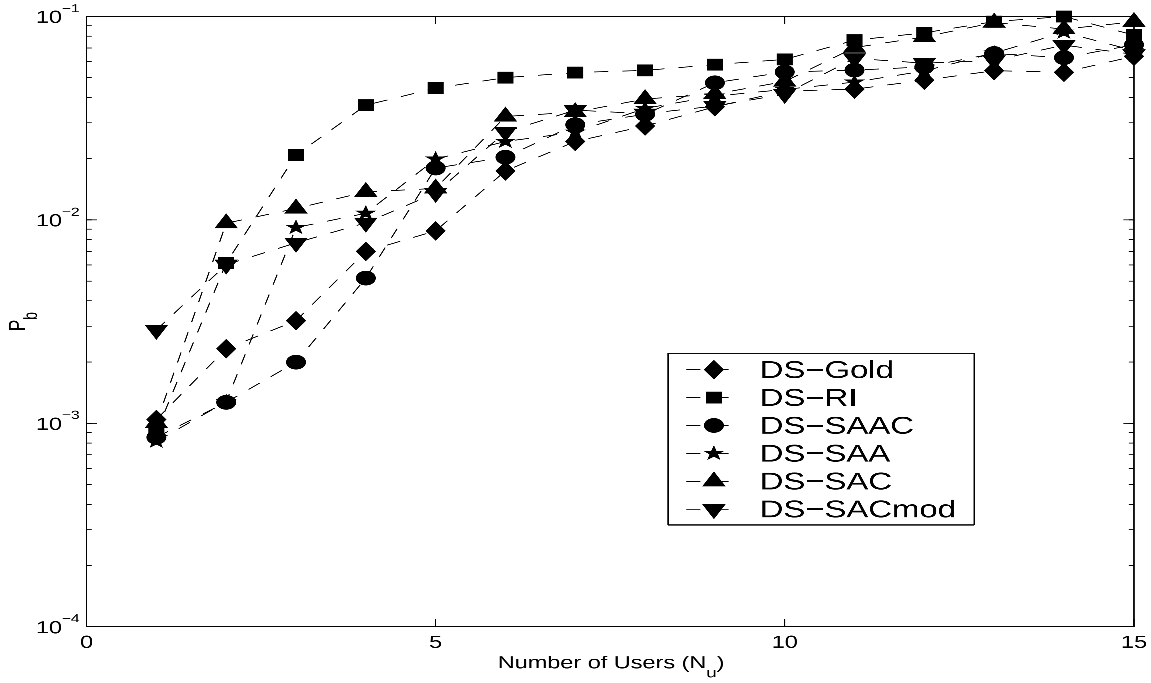


Figure 4.20 DS-BPSK MAI Results, $N_{MP} = 5$ Replicaions, Asynchronous Network

Table 4.12 Average TH-PPM BER Improvement(dB):Asynchronous Network

Code	Number of Multipath Replications (N_{MP})				
Family	0	2	5	10	40
RI	-4.07	-1.44	-0.69	-0.28	-0.03
SAAC	1.50	0.24	0.15	0.07	0.04
SAA	-2.67	-0.80	-0.36	-0.05	0.09
SAC	-1.40	-0.50	-0.21	-0.11	0.21
SACmod	-2.48	-0.76	-0.40	-0.20	0.08

Table 4.13 Average DS-BPSK BER Improvement(dB):Asynchronous Network

Code	Number of Multipaths (N_{MP})				
Family	0	2	5	10	40
RI	0.30	-3.40	-3.41	-2.84	-1.12
SAAC	0.32	-0.87	-0.28	-0.08	0.29
SAA	0.00	-1.05	-0.91	-0.72	-0.11
SAC	0.48	-2.41	-2.23	-1.90	-0.83
SACmod	0.08	-2.10	-1.59	-1.30	-0.53

4.7 Narrow Band Interference (NBI)

4.7.1 DSSS Interference: Offset in UWB Spectrum. The model for DSSS NBI is based on an IEEE 802.11 wireless transmission system. The interferer is based on IEEE 802.11b which uses three channels for Direct Sequence Spread Spectrum (DSSS) Differential Quadrature Phase Shift Keying (DQPSK) at 11 Mbps in the 2.4 to 2.48 GHz range (center frequency of each channel is separated by 25 MHz). Interference is injected directly into the demodulator, i.e., pre-filtering effects of the UWB antenna and RF filter are not included. Results for TH-PPM modulation (Fig. 4.21) indicate that degradation does occur, though not at the theoretical rate projected in (3.4). This indicates that concentrated interference power between 2.4 to 2.48 GHz degrades P_b performance less than equivalent interference power spread over the entire UWB signal bandwidth.

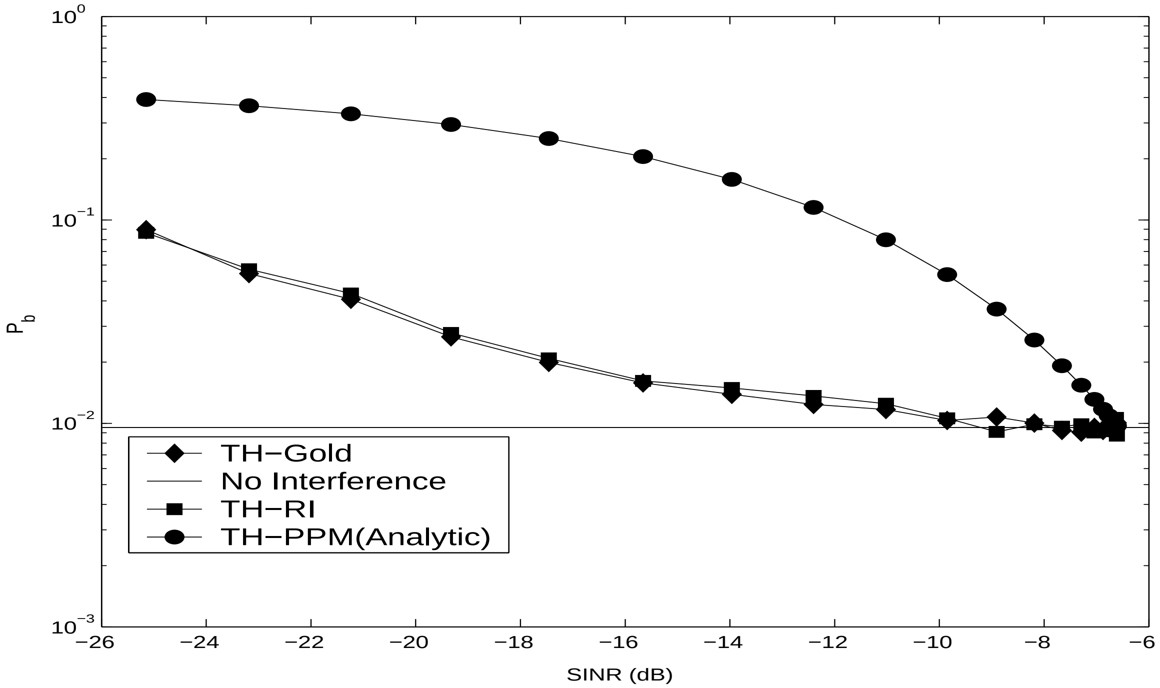


Figure 4.21 TH-PPM Offset DSSS Interference Results for $E_b/N_o = 7.5$ dB

Though all NBI tests are performed at $E_b/N_o = 7.5$ dB, DS-BPSK modulation is tested at a lower SINR ($10 \cdot log(N_c = 31) = 14.9$ dB lower) due to the inherent pulse integration gain associated with this form of signalling.

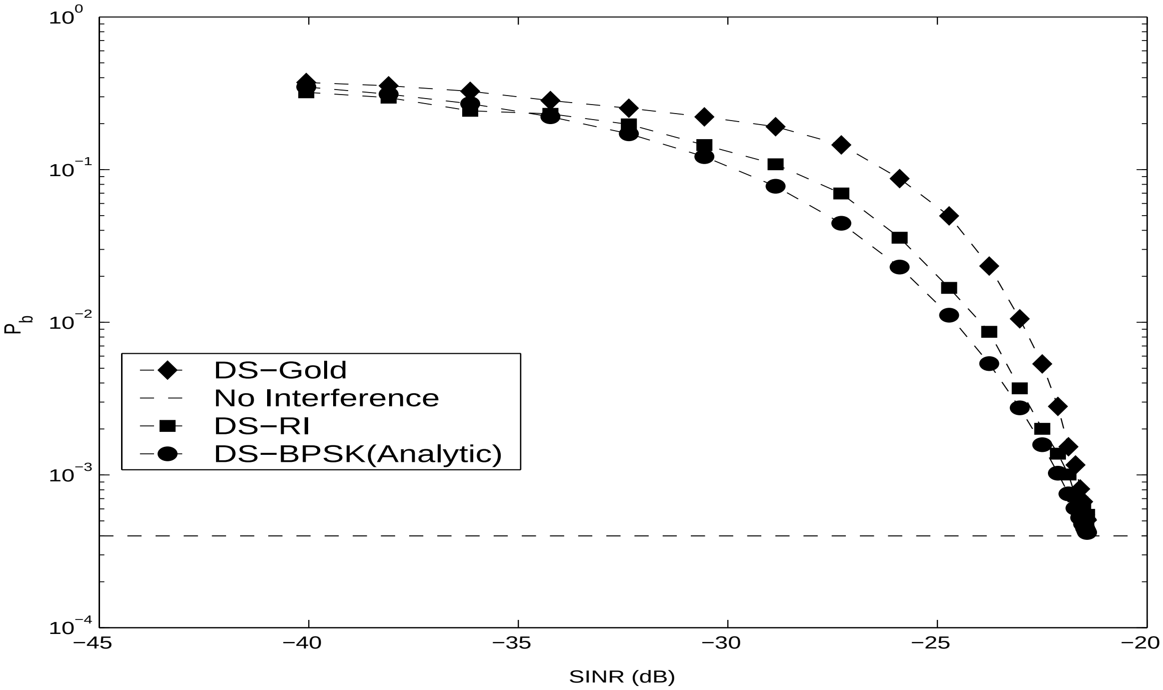


Figure 4.22 DS-BPSK Offset DSSS Interference Results for $E_b/N_o = 7.5$ dB

Analytic values of P_b for antipodal signalling are presented in Fig. 4.22 and represent a lower bound on BER performance. Data indicates that a DS-BPSK modulated UWB signal is more susceptible to concentrated DSSS interference power than the equivalent interfering power distributed across the entire UWB signal bandwidth.

4.7.2 DSSS Interference: Centered in UWB Spectrum. The second simulated interferer is identical to the first with the exception that the operating frequency is placed in the middle of the UWB waveform spectrum at 5.0 GHz. The SNR is again held constant at $E_b/N_o = 7.5$ dB while the interference power is varied to account for the SINR range displayed in Figs. 4.23 and 4.24. The horizontal line indicates P_b performance for the given SNR levels, with no interference present. Negligible difference was observed between the Gold and RI codes in TH-PPM. Gold coded pulses performed slightly poorer in DS-BPSK modulation. As interference power increases, both codes experience approximately 1.5 (TH-PPM) and 2.5 (DS-BPSK) orders-of-magnitude higher P_b levels, the expected performance degradation in the presence of NBI.

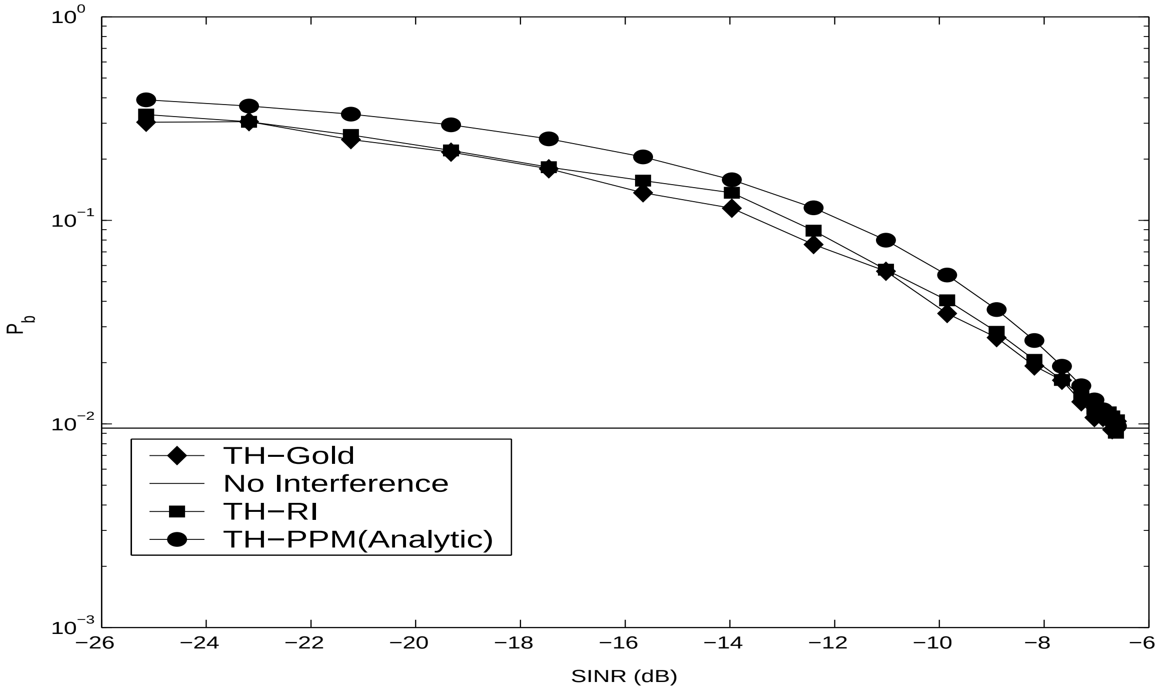


Figure 4.23 TH-PPM Centered DSSS Interference Results for $E_b/N_o = 7.5$ dB

4.7.3 FHSS Interference. The final interferer considered is based on Frequency Hopping Spread Spectrum (FHSS) IEEE 802.11. The system hopping rate of 10 hops/second was modified to 1 million hops every second to simulate a fast-hopping system in the presence of the relatively short UWB symbol duration. Simulations were performed at a fixed SNR level of $E_b/N_o = 7.5$ dB. Again, interference is injected directly into the demodulator without accounting for the effects of the UWB antenna and RF filter. Figures 4.25 and 4.26 display results of Gold and RI code performance in the presence of the FHSS interference ranging from 1.0 to 4.0 GHz in 10 MHz increments. The rate of P_b degradation is more varied in FHSS interference than DSSS interference due to the statistical nature of frequency hopping.

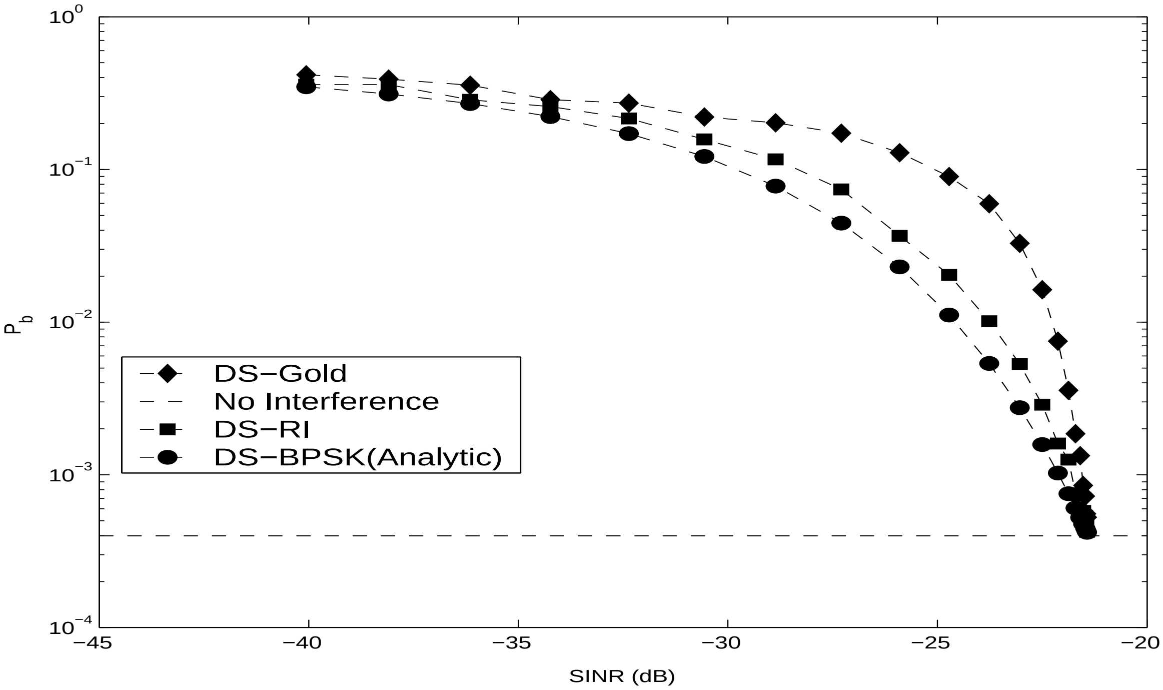


Figure 4.24 DS-BPSK Centered DSSS Interference Results for $E_b/N_o = 7.5$ dB

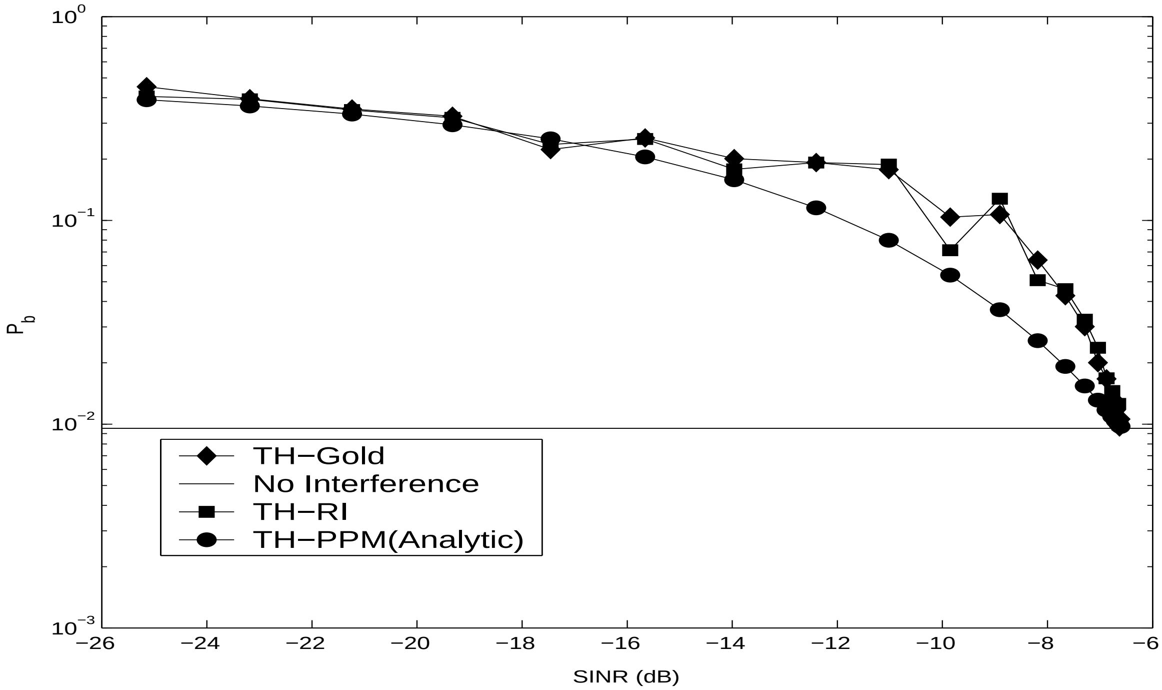


Figure 4.25 TH-PPM FHSS Interference Results for $E_b/N_o = 7.5$ dB

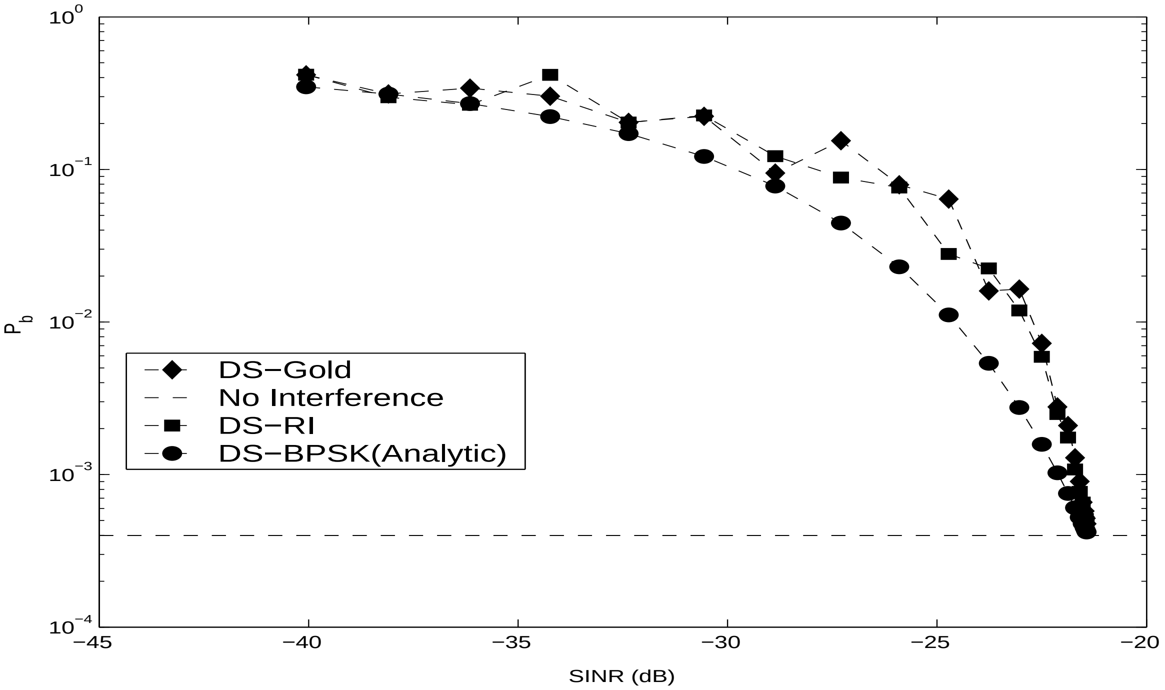


Figure 4.26 DS-BPSK FHSS Interference Results for $E_b/N_o = 7.5$ dB

4.8 Hybrid BPSK-PPM Modulation: Preliminary Investigation

4.8.1 Binary Hybrid Modulation. Communication performance is assessed for UWB signalling that combines PPM and BPSK modulation techniques. Binary hybrid signalling is achieved by (1) representing a “1” as a delayed (PPM) and 180^o phase shifted (BPSK) replica of the Gaussian waveform and (2) representing a “0” by the original Gaussian waveform. The binary hybrid modulation symbols are illustrated in Fig. 4.27 for one chip duration T_c.

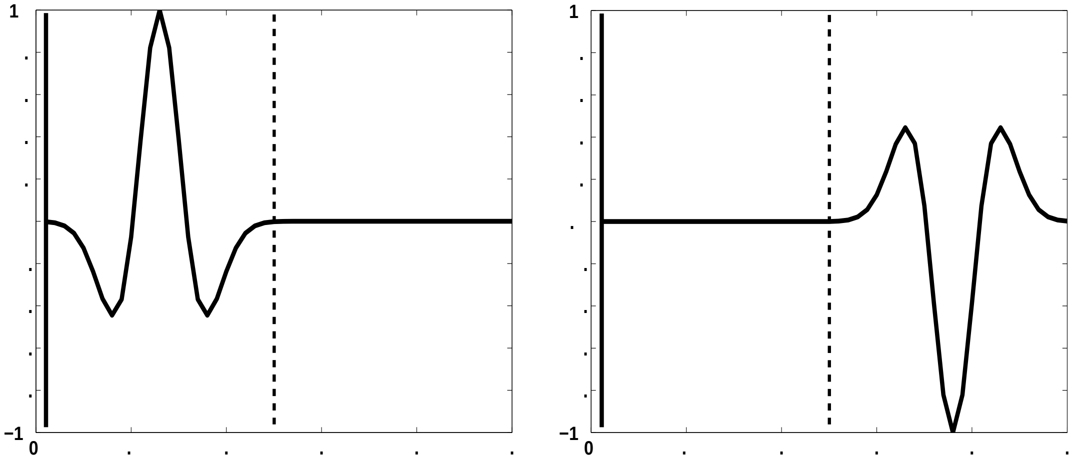


Figure 4.27 Binary Hybrid Waveforms: Left Position=“0” and Right Position=“1”

4.8.2 4-Ary Hybrid Modulation. Communication performance was measured for 4-Ary UWB signalling that combines PPM and BPSK modulations. Four signal values are represented in Fig. 4.28 by generating Gaussian waveforms in one of 2 positions (PPM) with one of 2 phase values (BPSK).

4.8.3 Hybrid Modulation Results. Figure 4.29, displaying Gold code results, demonstrates equivalent performance between binary hybrid modulation and theoretical orthogonal signalling.

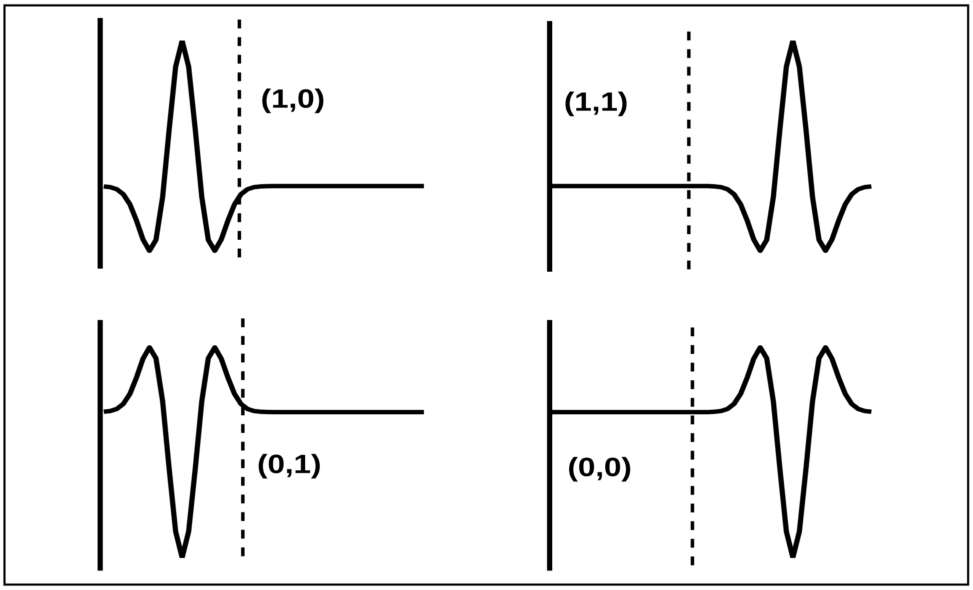


Figure 4.28 4-Ary Hybrid Waveforms: Bit Pattern (b_1, b_2)

Figure 4.29 displays Gold code results for 4-Ary signalling and theoretical binary antipodal signalling (BPSK). 4-Ary signalling doubles the data rate at the expense of a loss in E_b/N_o of 3 dB.

Figures 4.30 and 4.31 display results for Binary and 4-Ary Hybrid Modulations in both synchronous and asynchronous networks of 15 users. There appears to be negligible difference when compared to Gold coded TH-PPM modulation.

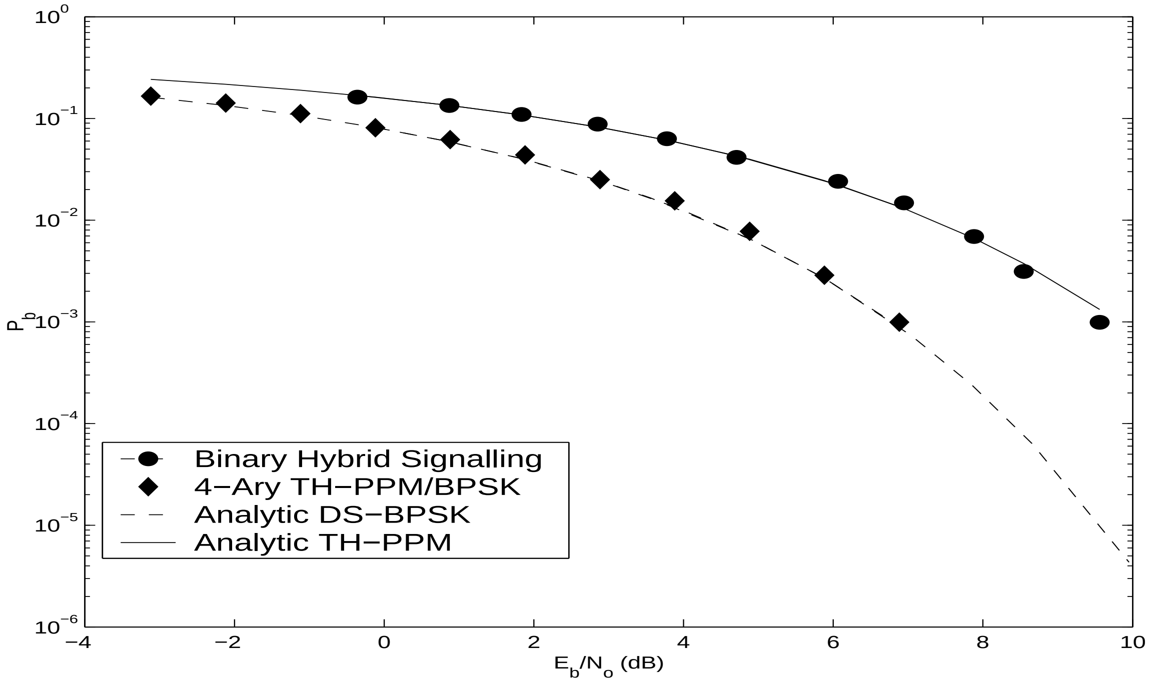


Figure 4.29 Binary and 4-Ary Hybrid Signalling With Gold Code length 31

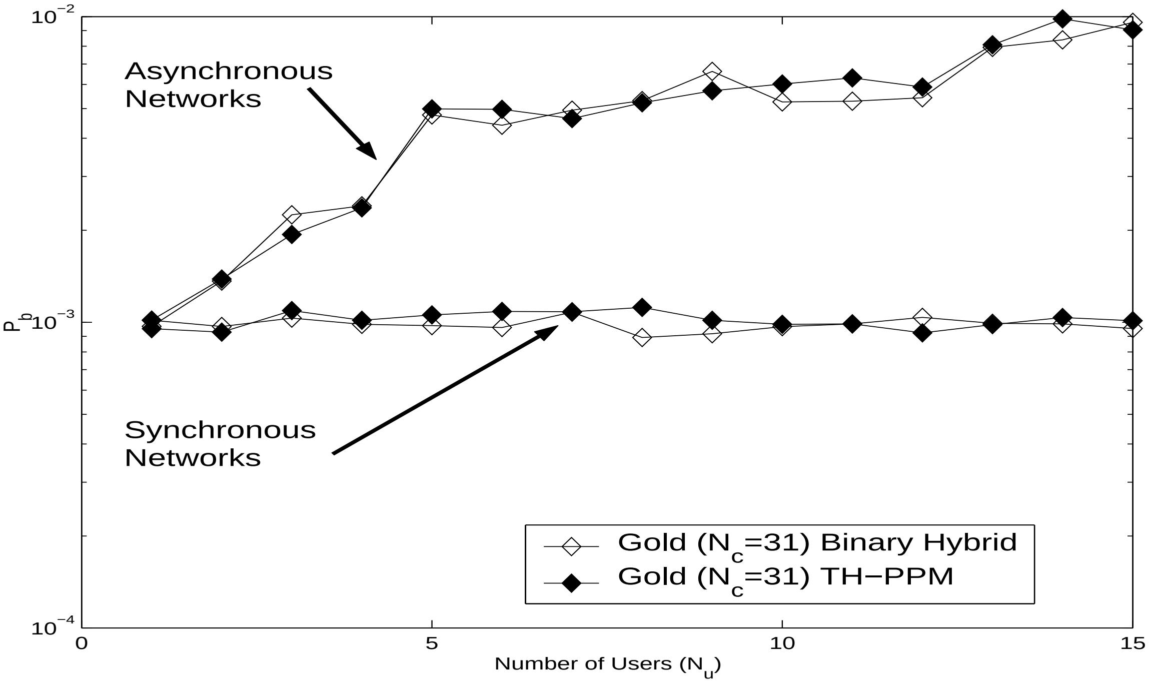


Figure 4.30 Gold Code Length 31 MAI Results for Binary Hybrid Signalling

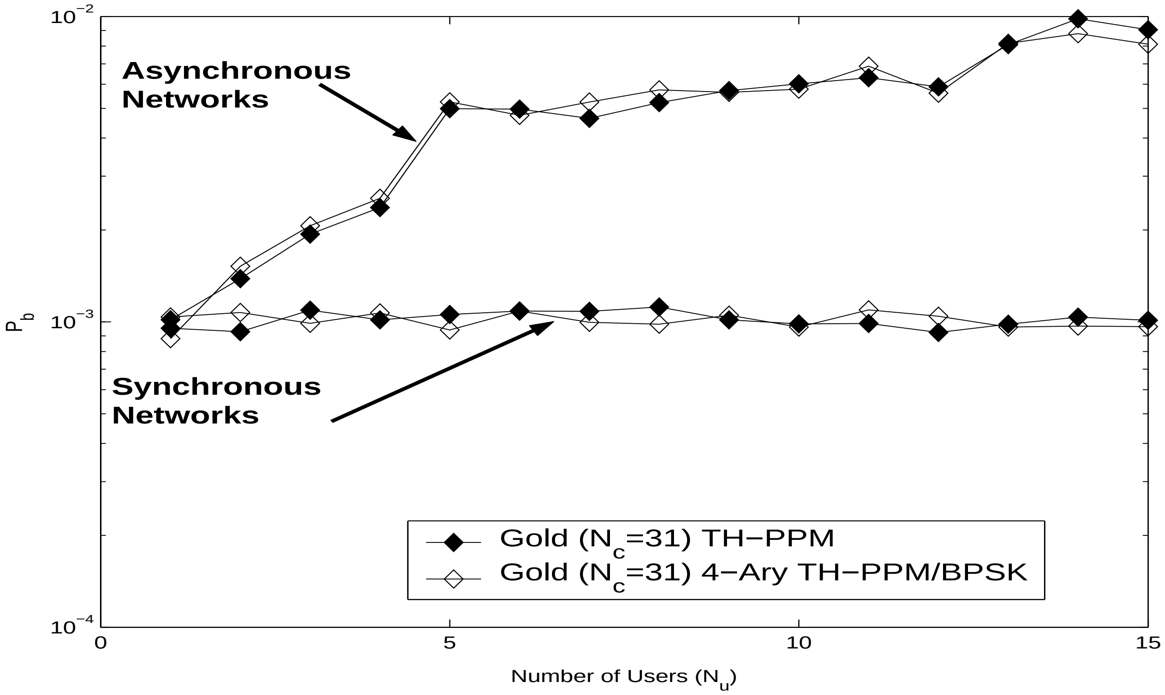


Figure 4.31 Gold Code Length 31 MAI Results for 4-Ary TH-PPM/BPSK Signalling

4.9 MAI and MPI Results for $N_c = 127$ Length Gold Codes

Results presented previously are simulated using codes of length $N_c = 31$. As shown in Figs. 4.32 and 4.33, additional results are obtained by simulating a synchronous network of 15 users for $N_{MP} = 0, 10$, and 40 replications using a Gold code of length $N_c = 127$.

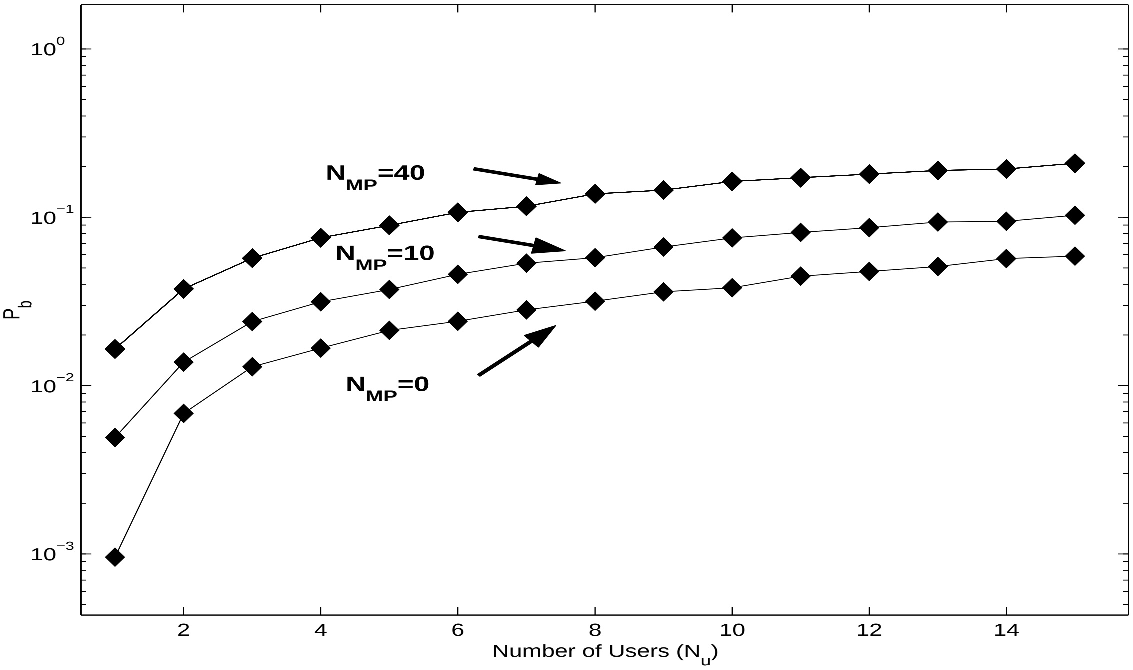


Figure 4.32 TH-PPM MAI Results, $N_{MP} = 0, 10, 40$ Replications, $N_c = 127$ Length Gold Code

In the TH-PPM case, the BER for the $N_c = 127$ length code is approximately two orders-of-magnitude higher (poorer) than the $N_c = 31$ length code (cf., Figs. 4.32 and 4.11). This is because no adjustments were made in the binary-to-decimal position generator that uses a register size of $b = 5$. Since the 127 length Gold code is created by a register size of $b = 7$ ($2^{(b=7)} = 128$), it is expected that the decimal position mapping is less optimal in the 127 length code. For $N_{MP} = 10$ and 40, there is negligible difference between the $N_c = 31$ and $N_c = 127$ cases.

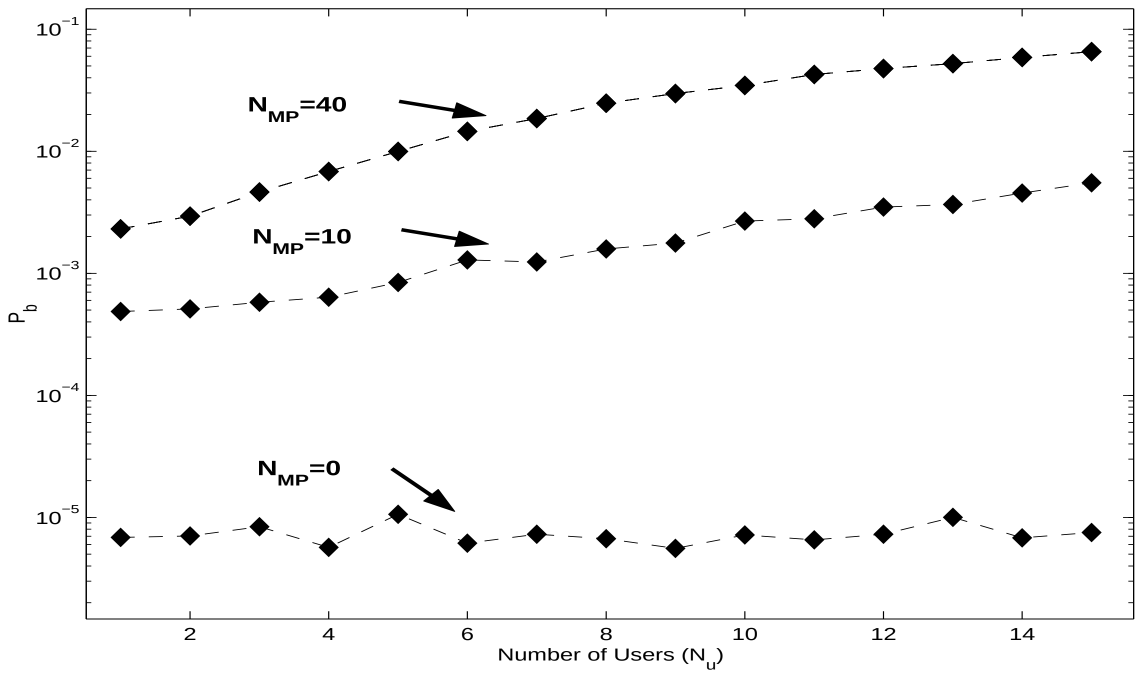


Figure 4.33 DS-BPSK MAI Results for MPI, $N_c = 127$ Length Gold Code

Using the $N_c = 127$ length code, DS-BPSK modulation exhibited an average improvement of 7.5 dB, 9.6 dB, and 4.6 dB for $N_{MP} = 0, 10,$ and 40, respectively. The improvement in BER derived from DS-BPSK is due to additional pulses used to represent a single bit of information (at the expense of decreased data rate). Over the three multipath levels tested, improvement averages 6.6 dB, nearly identical to the code length increase factor of $10 \times log_{10}(127/31) = 6.1$ dB.

V. Conclusions

5.1 Research Contributions

Analysis of multiple access (MA) communication performance using ultra wide band (UWB) waveforms with TH-PPM and DS-BPSK modulation techniques is provided. Results presented expand the UWB communication knowledge base, building upon previous UWB research involving Gold code spreading sequences. Contributions include UWB communication, multiple access interference (MAI), and multipath interference (MPI) performance characterization using Gold, Random Integer (RI), and Simulated Annealing (SA) codes over synchronous and asynchronous networks. Narrow band interference (NBI) effects are simulated to determine the impact existing wireless technologies may have on UWB system performance.

5.2 Summary of Findings

For a *synchronous network* containing up to 15 users, TH-PPM modulation with Gold coding provided average BER improvement factors of 0.019 (-17.1 dB) and 0.273 (-5.64 dB) over RI and equally weighted SA coding (SAAC), respectively. Likewise, DS-BPSK modulation with Gold coding provided average BER improvement factors of 0.013 (-18.9 dB) and 0.002 (-26.3 dB) over RI and equally weighted SA coding (SAAC), respectively. In a MA environment with strong MPI, TH-PPM performance was virtually independent of code selection and DS-BPSK performance was most robust when using Gold coding.

In an *asynchronous network*, there is less variation in BER. With no MPI present, the maximum P_b level for 15 MA interferers is an order-of-magnitude lower than in the synchronized network using TH-PPM and DS-BPSK modulations. However, in the asynchronous network, the best performing code in the synchronous case (Gold) increased by at least an order-of-magnitude in BER. In the asynchronous TH-PPM network, the SAAC code averaged the best performance. In the asynchronous

DS-BPSK network, the SAC code performed best with no MPI present and the SAAC code achieved the lowest BER average when strong MPI was present.

In general, BER varied linearly with the number of multipath replications (N_{MP}). For a single transmitter/receiver link, TH-PPM averaged a factor of 1.5 loss in P_b per multipath replication, or P_b (degraded)=$[1.5 \times N_{MP} + 1] \times (P_b \; at \; N_{MP} = 0)$ over a P_b range of $[1 \times 10^{-3} : 5 \times 10^{-2}]$ for $N_{MP} = [0 : 40]$ replications. Likewise, over the same multipath range, average DS-BPSK performance is given by P_b (degraded)=$[25 \times N_{MP} + 1] \times (P_b \; at \; N_{MP} = 0)$ over a P_b range of $[8 \times 10^{-6} : 1 \times 10^{-2}]$.

For almost all network configurations and modulation types tested, the interacting factors *Users*N_{MP}* contributed most to BER variation. The only exception was in the DS-BPSK synchronized network where *Users*N_{MP}* contributed 37.2% and *Code*Users* was responsible for 39.2% variation.

5.3 Future Research

5.3.1 M-Ary Signalling.

The modulation schemes and codes used here were biphase. There is no theoretical basis for limiting UWB signalling to binary. Simple modifications can be made to assess the performance of DS M-Ary Phase Shift Keying (DS-MPSK) and TH M-Ary Pulse Position Modulation (TH-MPPM) by using multiple phase and symbol offsets within a chip interval. Some researchers have reported on a potential benefit using overlapping pulses [34].

5.3.2 UWB Detection Using a Narrow Band Receiver.

It may be possible to non-cooperatively detect and/or intercept a UWB transmission using cyclostationary signals with a narrow band receiver [15],[16]. In the case of TH-PPM UWB signalling for instance, pulses are repeated periodically and pseudo-randomly shifted during specific intervals. If it is possible to make the effects of this time dithering negligible, the UWB waveform repetition could be used to discover the signal.

5.3.3 Coexistence with Wireless Technologies. Compatibility and coexistence issues abound with various wireless protocols in existence today. A survey of existing protocols and their interaction could be further investigated. The narrow band testing reported herein is an initial look at UWB interference. More detailed, realistic models could be developed to more closely mimic actual systems. For instance, no power adjustments were made in the interference tests. Rayleigh channel fading is a potential model adjustment that could be made.

In addition to interference effects, encryption and data security requirements of wireless communications must be addressed to make UWB communication mediums practical and reliable. The capability for this type of research exists at several institutions [6].

5.3.4 Channel Models. An AWGN channel was assumed in this study. The validity of this assumption is questionable and is the subject of future study [7]. No fading channel effects were implemented and all multipath signals were received with equal power. As mentioned in Section 2.4.2, fading channels such as Rayleigh and $\triangle - K$ appear to be reasonable channel models to incorporate into UWB models.

5.3.5 Parameter Adjustment. The data rates used for simulation were fixed. Varying the data rate could have a significant impact on communication performance. An effort could be undertaken to determine optimal pulse spacing and chip size while maximizing efficiency and avoiding overspreading effects described in Section 2.4.2. In the case of TH-PPM modulation, the code-to-position mapping algorithm could be tested to determine optimal register size and shift.

5.3.6 UWB Technological Implications. By allowing unlicensed UWB operation in the 3.1 to 10.6 GHz range, the FCC has opened the door for implementing newer versions of older technologies. Modifying older technologies to operate in this new unlicensed spectrum may lead to increased user capacity and interference suppression.

Appendix A. Additional Results

A.1 TH Code Metrics

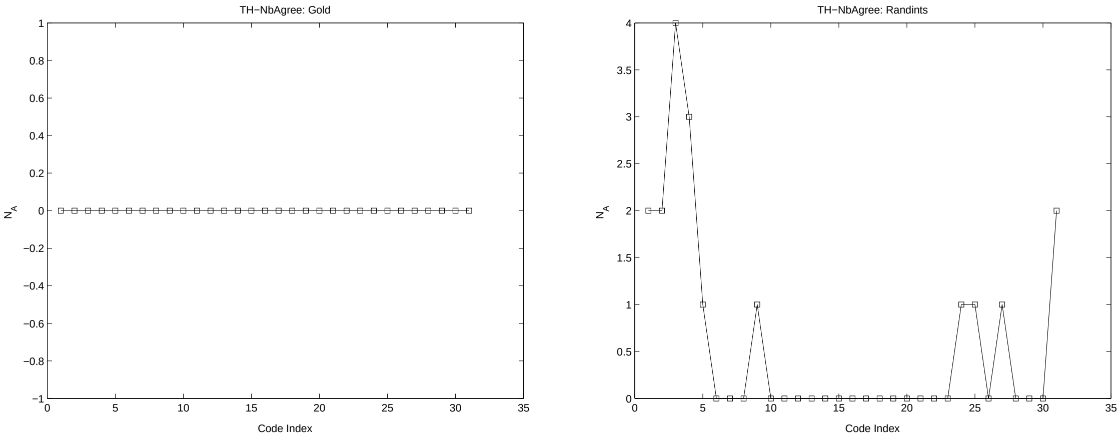


Figure A.1 TH # Collisions/Time Slot: Gold (Left) and RI (Right)

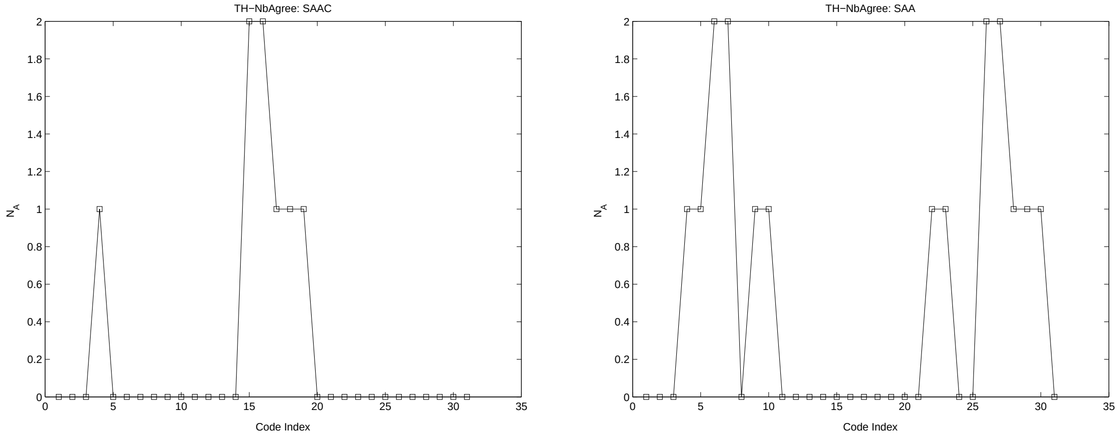
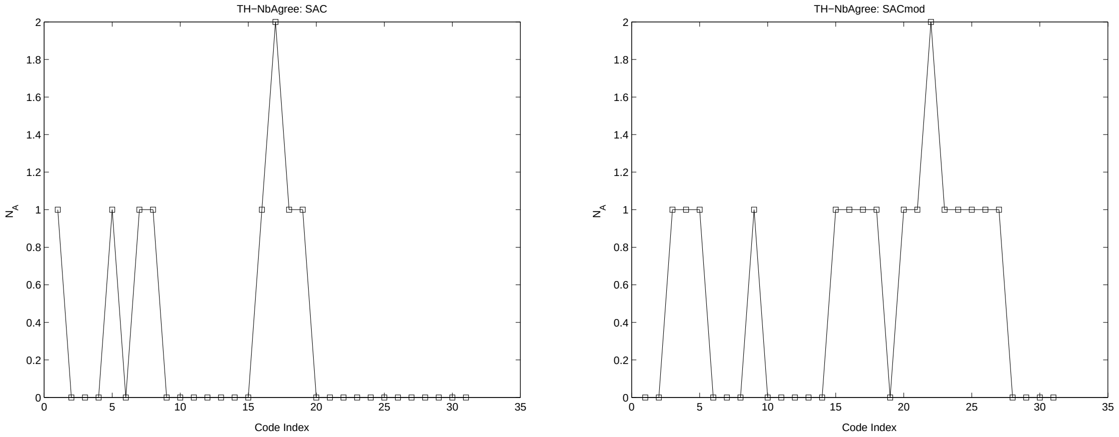


Figure A.2 TH # Collisions/Time Slot: SAAC (Left) and SAA (Right)


TH−NbAgree: SAC

TH−NbAgree: SACmod

Figure A.3 TH # Collisions/Time Slot: SAC (Left) and SACmod (Right)

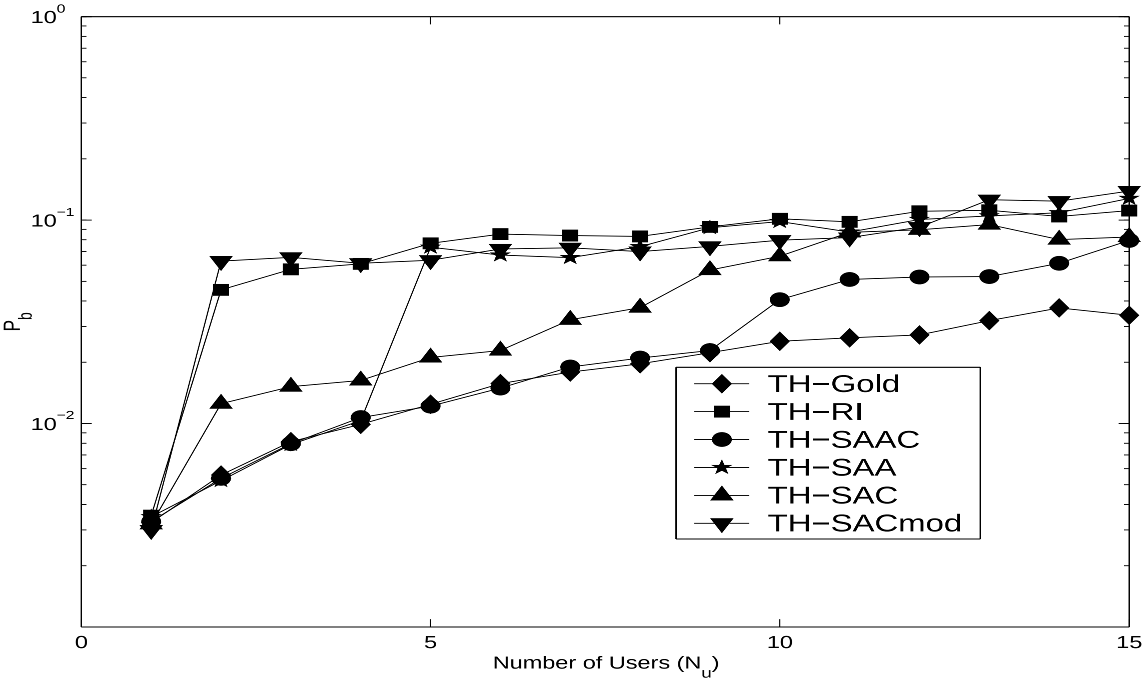


Figure A.4 $N_{MP} = 2$ MAI Results: Sync. TH

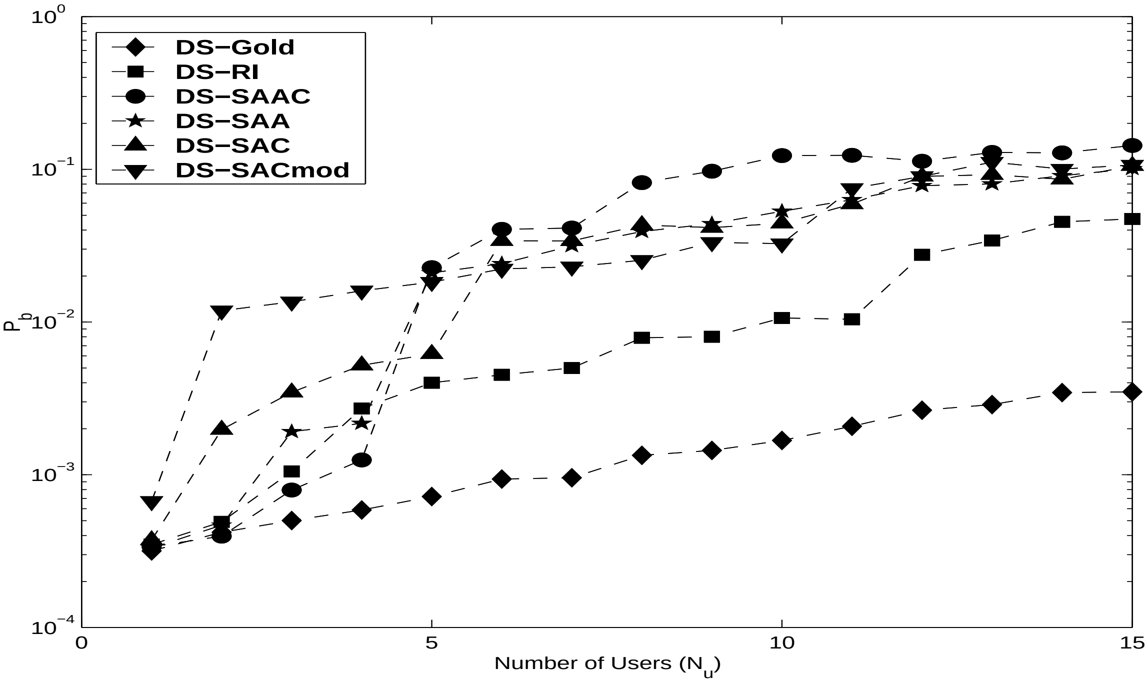


Figure A.5 $N_{MP} = 2$ MAI Results: Sync. DS

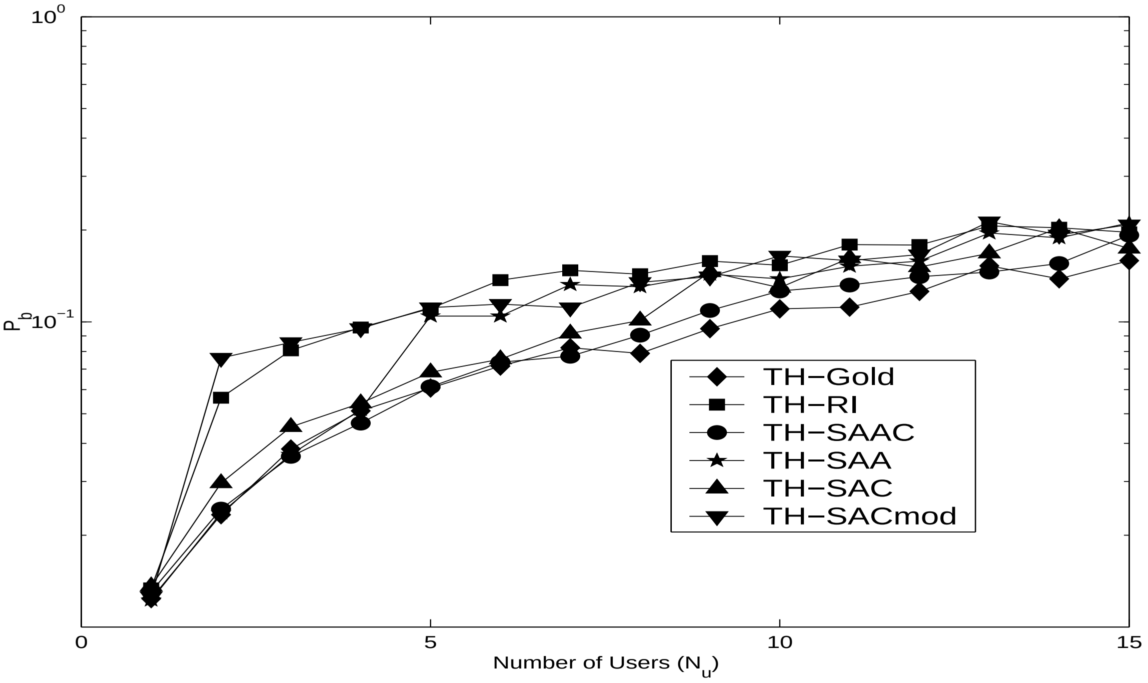


Figure A.6 $N_{MP} = 10$ MAI Results: Sync. TH

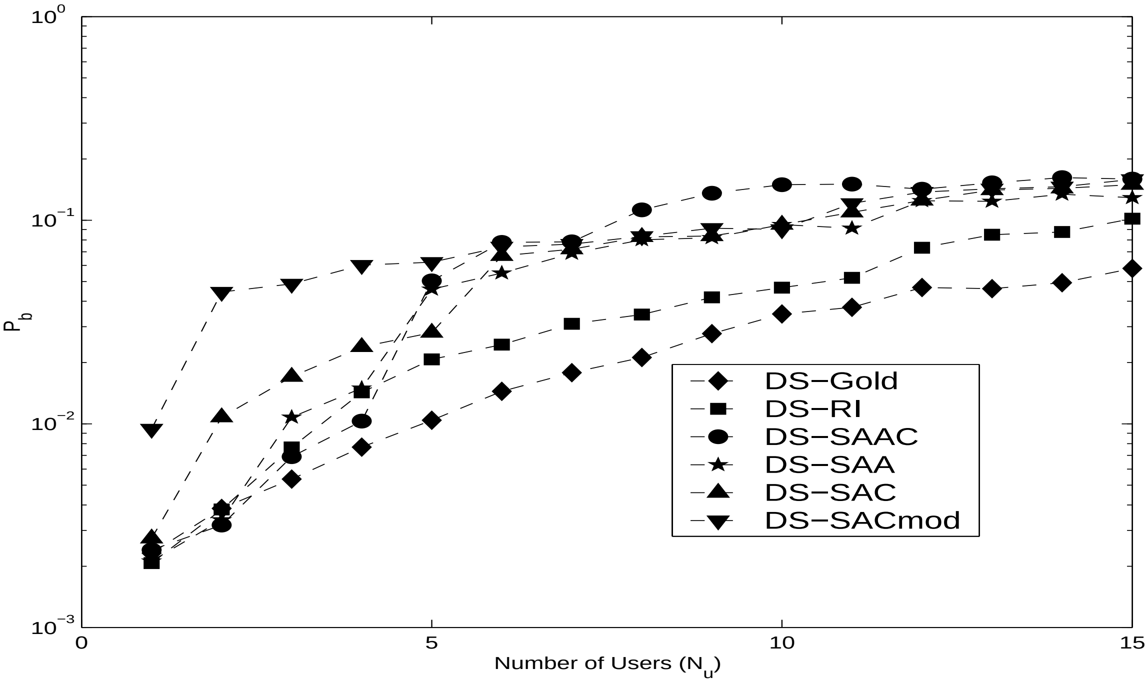


Figure A.7 $N_{MP} = 10$ MAI Results: Sync. DS

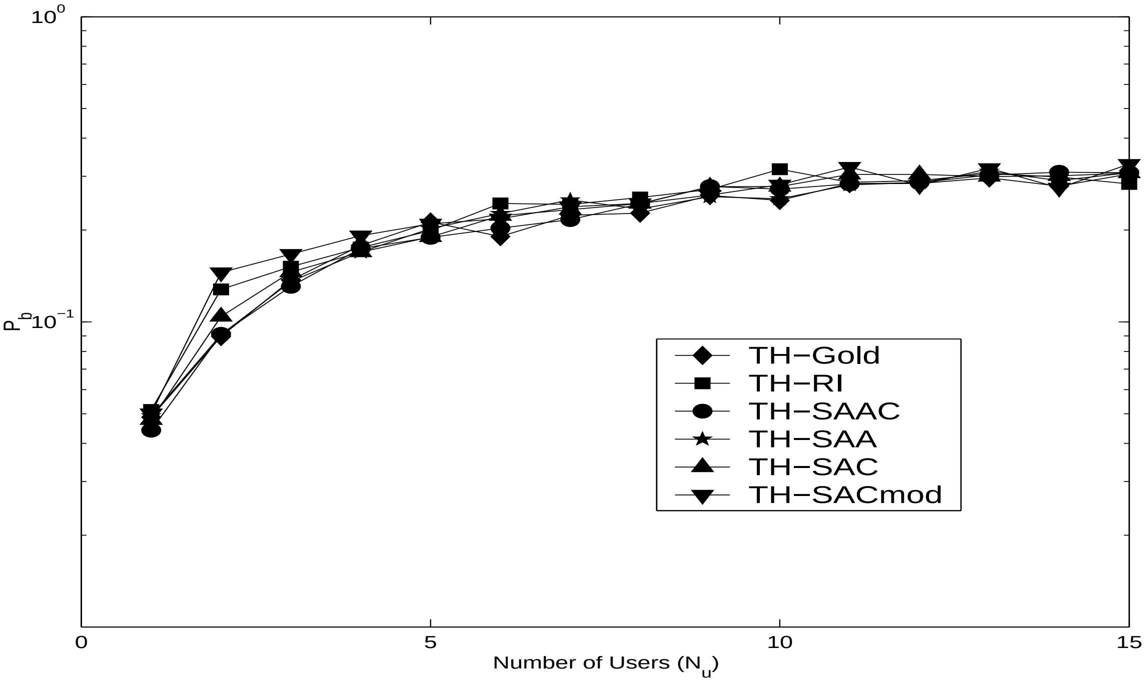


Figure A.8 $N_{MP} = 40$ MAI Results: Sync. TH

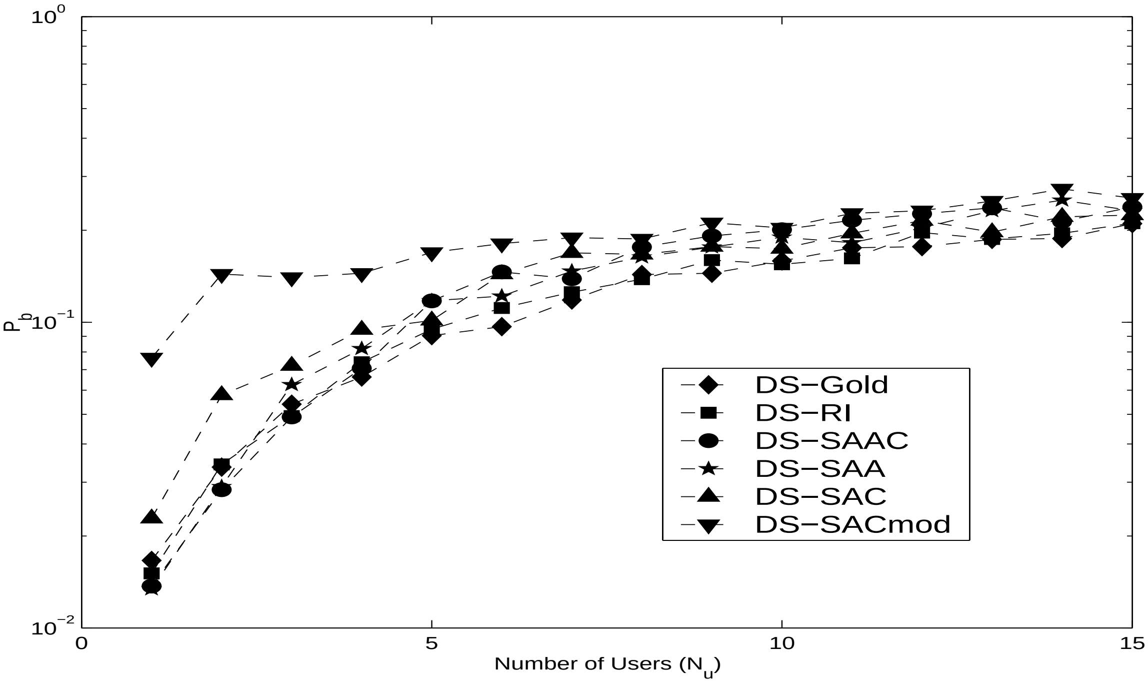


Figure A.9 $N_{MP} = 40$ MAI Results: Sync. DS

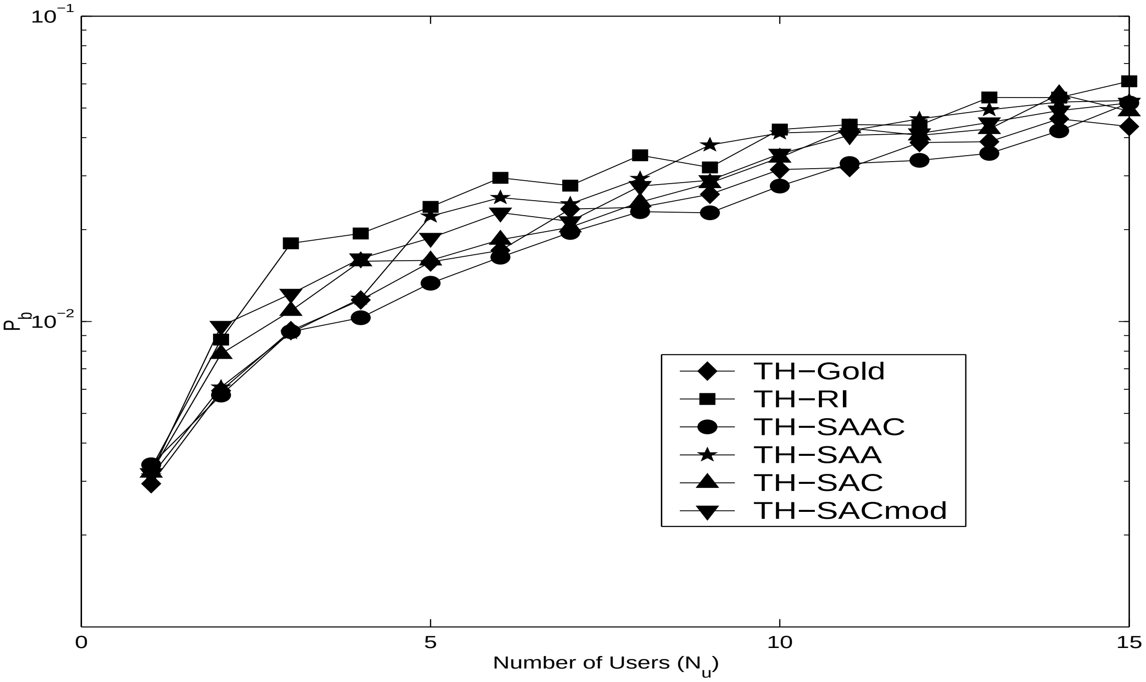


Figure A.10 $N_{MP} = 2$ MAI Results: Async. TH

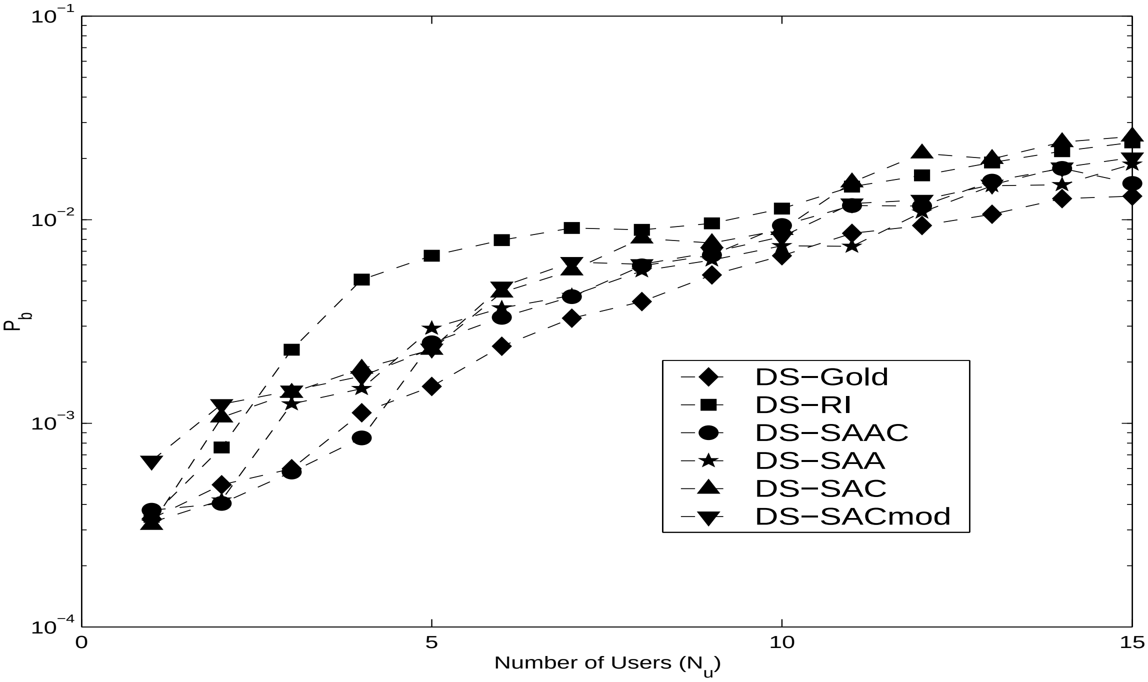


Figure A.11 $N_{MP} = 2$ MAI Results: Async. DS

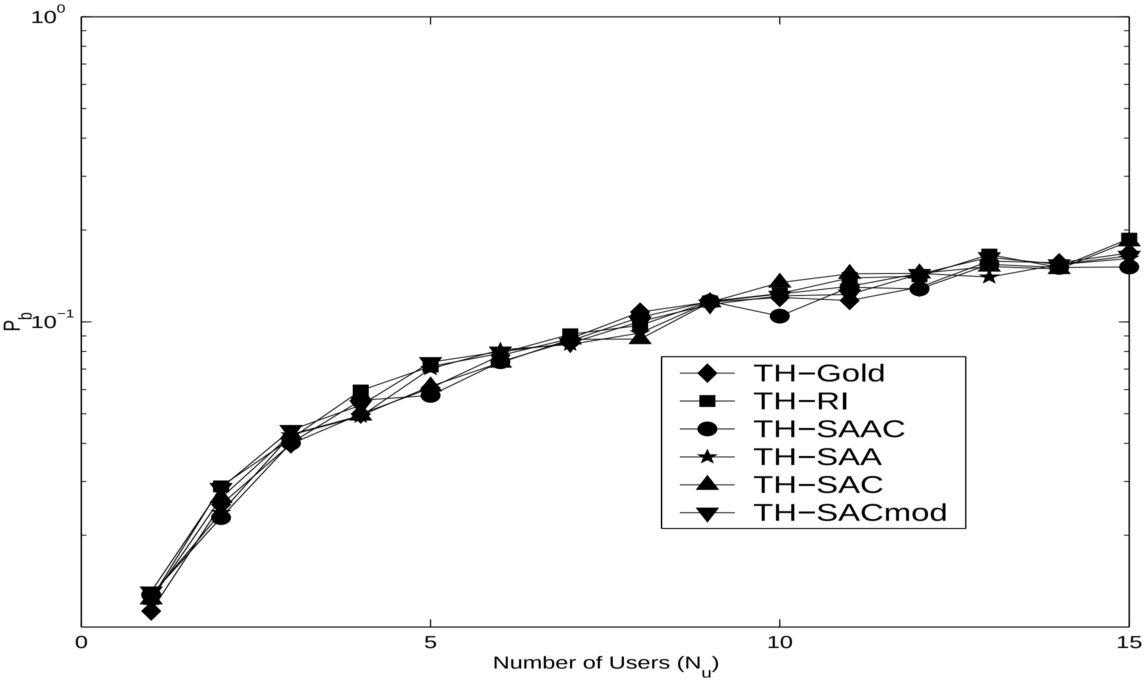


Figure A.12 $N_{MP} = 10$ MAI Results: Async. TH

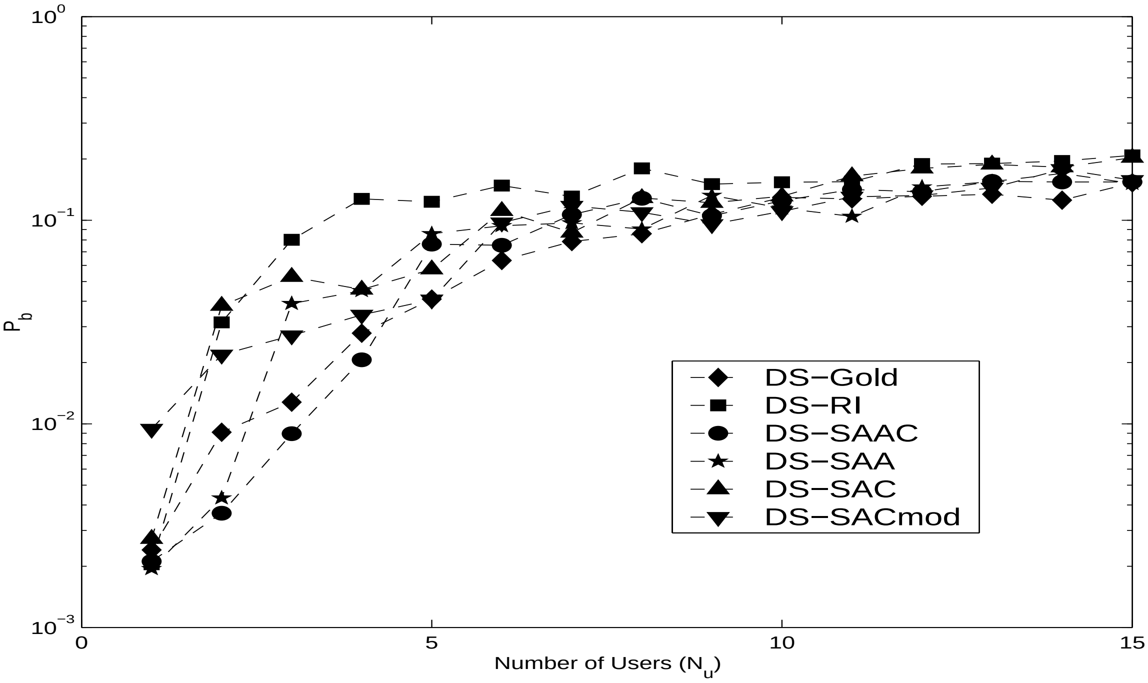


Figure A.13 $N_{MP} = 10$ MAI Results: Async. DS

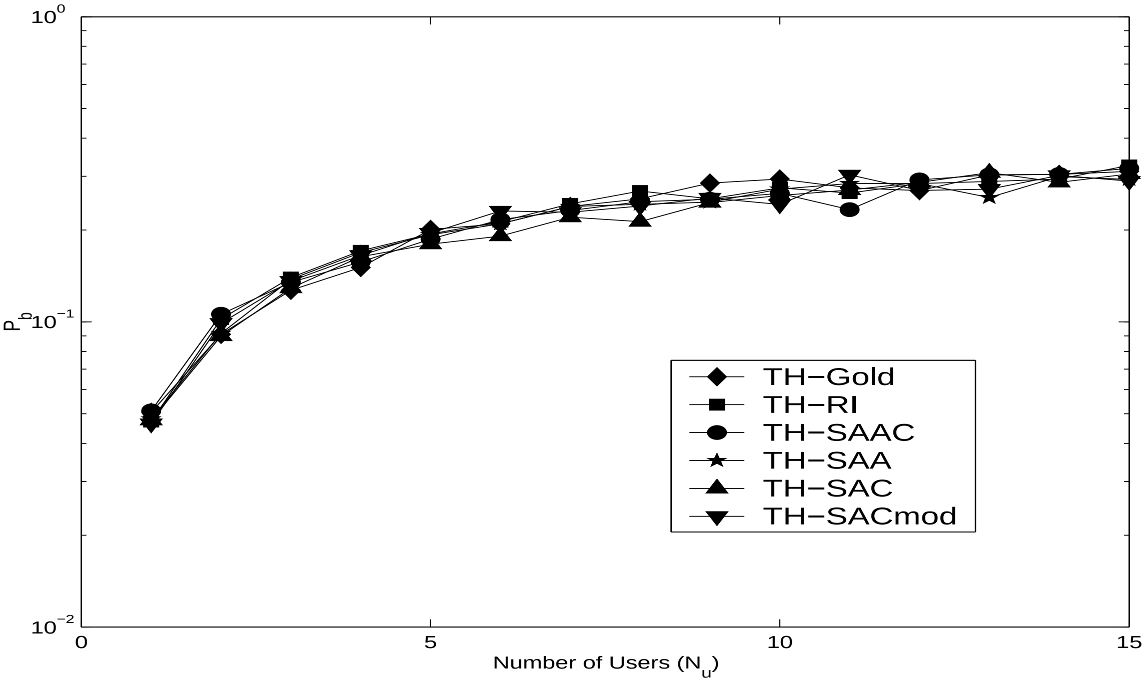


Figure A.14 $N_{MP} = 40$ MAI Results: Async. TH

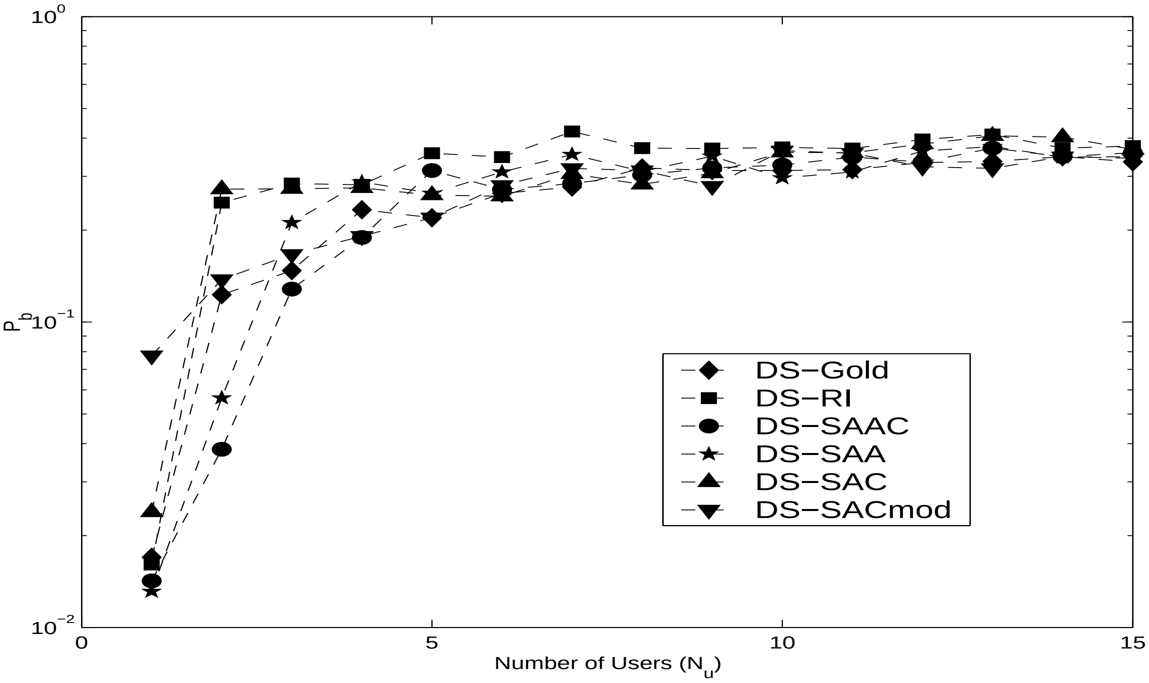


Figure A.15 $N_{MP} = 40$ MAI Results: Async. DS

Appendix B. Simulation Code

The Matlab® code developed for simulations is reproduced here. The primary code used is function mainuwb.m listed immediately below. Supporting non-standard functions codeselect.m, dataplot.m, cmc_codestat.m, and agreeSeq.m are also attached.

B.1 mainuwb.m

```
function mainuwb(SNRdB,codeselector,nb_users,...
    var_looplim,mp_num,modul_flag,NBI_flag,sync_flag)
%
%function mainuwb(SNRdB,codeselector,nb_users,var_looplim,mp_num,modul_flag,NBI_flag,sync_flag)
%Eg: mainuwb([-19:1:-19],2,15,3,5,0,2,1)
%
%SNRdB       =Avg power SNR (Nc=31 then SNRdB=-19dB->Eb/No=10dB)
%Codeselector=code chosen for users. See codeselect.m
%nb_users    =Nb MAI transmitters plus desired signal
%var_looplim =Nb trials of entire experimental setup run
%mp_num      =Nb replicated signals transmitted in addition to each user
%modul_flag  =0 for TH-PPM and DS-BPSK, 1 turns off TH-PPM, 2 turns off DS-BPSK
%NBI_flag    =0 for no NBI, 1 for 802.11b NBI model
%sync_flag   =0 for synchronous network of users,1 for asynchronous users
%
%Running this function without arguments defaults to plotdata.m which
%will produce graphs of the waveform in time and frequency domains.
%
%mainuwb.m
%Author:        2LT Courtney M. Canadeo
%Created:       August 2002
%Last modified: February 2003
%
%Output graphs: Pb vs SNR,Users
%Non-standard required functions:
%   codeselect.m    gold_gen.m  oct2bin.m
%   mSeqGen.m       dataplot.m
%Additional/optional function calls:
%   testcases.m     dataload.m  analyze.m
%   cmc_codestat.m  agreeSeq.m
%When save_flag==1, data and graphs are stored in subdirectories /data and /results
%respectively. This information can be opened manually or by running scripts
%dataload.m, figload.m, or analyze.m

%When running as a function from testcases.m:
%--global parameters section is commented
%--save_flag=1
%--Comment "clear all"

%clear all;
close all;tstart=clock;

show_analysis   =1;               %1=Draw graphs and save data
save_flag       =1;               %1=Save data and graphs

%SNRdB=-18.9 for Pb~7e-6 when Gold 31 since 15dB:SNRc->SNRu & 14dB:SNRu->EbNo
%SNRdB=-3.99 for Pb~5e-6 when No code (10dB=Eb/No)
%Define Global Parameters:
if nargin==0
    save_flag=0;                  %Do not save if no arguments passed
    SNRdB=[-49:1:-19];            %Enter begin loop SNR value in dB
    codeselector=8;               %Ref codeselect.m: 1=NoCode 2=Gold31 3=SA31 8=Randints
    nb_users=5;                   %Number users for MAI testing
    var_looplim=1;                %Number variance runs
    mp_num= 0;                    %Number multipaths | ="0":disable multipath
    modul_flag=0;                 %2=Turn off DS | 1=Turn off TH | 0=Both on
    NBI_flag=2;                   %1=Turn on NBI; SNRdB->SJRdB & SNR->Pb=1e-5
    sync_flag=0;                  %0=synchronous network of users | 1=async
```

```
end
%******************************Above this line, parameters often varied
nb_usersbeg=1;                %Number users begin testing | ="nb_users":disable Loop
row=124;                      %Nb pulses/loop,optimized: speed|accuracy|memory|code periodicity
Twp=.08e-9;                   %Pulse width parameter: width T~2.5*Twp
T =.2e-9;                     %Pulse duration
dt=Twp/10;                    %A/D sampling resolution
To=15e-9;                     %Pulse repetition interval
Ts=0;%=12.4ns,12.8ns (DS,TH)  %Symbol duration (PRI) defined by code below
Tc=(2*T/dt);                  %Chip duration
u=0:dt:(To-dt);               %Time for single pulse
loop_ber=length(SNRdB);       %Determine nb of SNR trials
loop_reset=loop_ber;          %Reset for multiple Pulse trials
randn('state',sum(100*clock));%Increase 'integrity' of randomness

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%TRANSMITTER%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%PULSE GENERATOR%%%%%%%%%%%%%%%%%%%%%%%
%Generate Gaussian Monocycle Pulsetrain:
%Vt=-2*(u-1.3*Twp)/T^2.*exp(-2*((pi*(u-1.3*Twp))/T).^2);             %1st deriv.
Vt=(1-4*pi*((u-1.2*Twp)/Twp).^2).*exp(-2*pi*((u-1.2*Twp)/Twp).^2);  %2nd deriv.
lenVt=length(Vt); pultr=repmat(Vt(1:Tc),row,1);

%GENERATE CODE FOR MULTIPLE ACCESS (MA) & Define symbol duration:
[code,codetype,lenc]=codeselect(codeselector);
lencds=Tc*lenc;
lencth=Tc*(lenc+1);

%%%CREATE SIGNAL REFERENCES:%%%
d=Tc/2; ref_bpsk1= Vt(1:Tc); ref_bpsk0=-Vt(1:Tc);
ref_ppm0=Vt(1:Tc); ref_ppm1(1:Tc)=zeros(1,Tc);
ref_ppm1(d+1:Tc)=Vt(1:d); ref_matrix=[ref_bpsk0; ref_bpsk1;
ref_ppm0; ref_ppm1];

%%%TH CODE GENERATION%%%
if modul_flag~=1
    if codeselector==1                  %NoCode case
        th_ck(1:nb_users,1:row)=0;      %randint(nb_users,row,(lenc+1))*Tc;
    else
        th_ck=zeros(nb_users,lenc);
        for j=1:nb_users
            for i=1:lenc
                regsz=5;
                r=mod(i-1,lenc);
                if r<=(lenc-regsz)
                    th_ck(j,i)=bin2dec(sprintf('%d',code(j,r+1:r+regsz))) *Tc;
                else
                    th_ck(j,i)=bin2dec(sprintf('%d%d',code(j,r+1:lenc),code(j,1:r+regsz-lenc))) *Tc;
                end
            end
        end
   end
   th_ck(1:nb_users,1:lenc*row)=repmat(th_ck(1:nb_users,1:lenc),1,row);
    %%%Reference values for TH codes, if using TH/DS-PPM
    %     for j=1:nb_users
    %         ref_ppm0_code(j,:)=0;
    %         for k=1:lenc
    %             if code(j,k)==0
    %                 ref_ppm0_code(j,k*Tc-Tc+1:k*Tc)=[ref_ppm0];
    %             else
    %                 ref_ppm0_code(j,k*Tc-Tc+1:k*Tc)=[ref_ppm1];
    %             end
    %         end
    %         ref_ppm1_code(j,:)=zeros(1,lencth);
    %         ref_ppm1_code(j,d+1:lencth)=ref_ppm0_code(j,:);
    %     end
end

%%%DS CODE GENERATION%%%
if modul_flag~=2
    for j=1:nb_users
        ref_bpsk0_code(j,:)=0;
        for k=1:lenc
            if code(j,k)==0
                ref_bpsk0_code(j,k*Tc-Tc+1:k*Tc)=[ref_bpsk0];
            else
                ref_bpsk0_code(j,k*Tc-Tc+1:k*Tc)=[ref_bpsk1];
            end
        end
        ref_bpsk1_code(j,:)=-1*ref_bpsk0_code(j,:);
    end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%END PRELOAD CONFIGURATIONS%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%---LOOP1:  %Variance Runs Begin
for var_loop=1:var_looplim
%---LOOP2   %RUN CODE FOR VARIOUS NUMBER OF USERS:
    pass_lim=nb_users;
    for pass_user=nb_usersbeg:1:pass_lim     %INITIALIZATIONS FOR LOOPS/COUNTERS:
        nb_users=pass_user;                 nberrors=0;     pass_pls=1;
        dispflag=0;                         SNRcheckflag=1; qval=0;     nbk=1;
        ds_error_flag(1:loop_reset)=1;      th_error_flag(1:loop_reset)=1;
        ds_repeat(1:loop_reset)=1;          th_repeat(1:loop_reset)=1;
        ds_errors(1:loop_reset)=0;          th_errors(1:loop_reset)=0;
        ds_symbols(1:loop_reset)=0;         th_symbols(1:loop_reset)=0;
        ds_demod_rx(1:row,1:loop_ber)=nan;  th_demod_rx(1:row,1:loop_ber)=nan;
        errcutds(1:loop_reset)=11;          errcutth(1:loop_reset)=11;
        %Determine which modulation scheme testing (DS or TH):
        %'emp' designates empirical value for eb(nb of errors)=300 in all cases
        %which is defined in code in the receiver
        if modul_flag==1
            th_errors(:)=999; modul='DSempXX';
            th_ber(1:loop_reset)=0;th_ber_user=0;
            th_error_flag(1:loop_reset)=0;
            th_repeat(1:loop_reset)=0;
        elseif modul_flag==2
            ds_errors(:)=999; modul='THempXX';
            ds_error_flag(1:loop_reset)=0;
            ds_repeat(1:loop_reset)=0;
            ds_ber(1:loop_reset)=0;ds_ber_user=0;
        else
            modul='BothempXX';
        end
        mywaitbar=waitbar(0,['Finding errors for variance pass ',num2str(var_loop),' and ',...
                num2str(pass_user),' user(s)...']);

%---LOOP3   %BEGIN LOOP FOR MULTIPLE PULSE MATRICES TRIALS:
        %Continue to input more pulses until all BERs found
        while((max(ds_error_flag) | max(th_error_flag))>0)
            waitbar(nberrors/300,mywaitbar)
            %Reset matrices in loop:
            ds_users_tx=zeros(row,lencds);
            th_mod_tx=zeros(row,lencth);
            th_users_tx=zeros(row,lencth);
            cos_matrix=zeros(row,lencth);
            begc=zeros(nb_users,row);tempbegc=zeros(nb_users,row);
            endc=zeros(nb_users,row);tempendc=zeros(nb_users,row);
            data_source1=randint(row,1);%Same data for user1 in DS & TH

            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %%%TX: DS-BPSK%%%%%%%%%%%%%%%%%%%%%
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            if(max(ds_error_flag)==1)
                if mp_num~=0                             %Setup Multipath values if needed:
                    mpo=15.4e-9/dt*randn(nb_users,mp_num*row);  %Match Foerster2002 results
                end
                data_source(:,2:nb_users)=randint(row,nb_users-1);data_source(:,1)=data_source1;
                if sync_flag==0
                    async=0;%%%async=0: PERFECT SYNCHRONIZATION for USERS
                elseif sync_flag==1
                    async=Tc*(lenc-1); %Use this if want multiples: =(Tc/d*lenc-1);
                end                   %w/ d*randint() below
                dso(2:nb_users)=randint(1,nb_users-1,async);dso(1)=0;%ds_user_offset:async. amongst diff. users
                if codeselector==1          %NoCode case
                    dso(2:nb_users)=randint(1,nb_users-1,25*Tc);
                    dso(find(dso>Tc))=d+1;  %Account for offset above Tc to
                end                         %not interfere, match Foerster2002 spacing
                for j=1:nb_users
                    ds_mod_tx=zeros(row,lencds);
                    for i=1:row
                        if data_source(i,j)==0
                            ref_bpsk01_code=ref_bpsk0_code;
                        else
                            ref_bpsk01_code=ref_bpsk1_code;
                        end
                        ds_mod_tx(i,1+dso(j):lencds)=ds_mod_tx(i,1:lencds-dso(j))+...
                            ref_bpsk01_code(j,1:lencds-dso(j));
                        %18 Feb. 03: added "ds_mod_tx(i,:)+" to the above line and
```

```
%            added next 'if' lines to check effects of wrapping delayed MAI into next
%            series of pulses to simulate periodic interference as opposed to aperiodic
if (i<row & j>1)
    ds_mod_tx(i+1,1:dso(j))=ref_bpsk01_code(j,1+lencds-dso(j):lencds);
    %DEBUG:
    %if (i>1 & i<row)
    %    figure(13)
    %    subplot(3,1,1)
    %    plot(ds_mod_tx(i-1,:));
    %    subplot(3,1,2)
    %    plot(ds_mod_tx(i,:));
    %    subplot(3,1,3)
    %    plot(ds_mod_tx(i+1,:));
    %    disp(['i= ' num2str(i) ' Paused...']);pause
    %end
end
for mp_loop=1:mp_num    %MULTIPATH added from each user/pulse
    rowOffset=fix((mpo(j,((i-1)*mp_num+mp_loop)))/lencds);
    sampOffset=round(rem(mpo(j,((i-1)*mp_num+mp_loop)),1)*lencds);
    if (i+rowOffset>0 & i+rowOffset<=row)
        if ((1+dso(j)+sampOffset)>0 & (dso(j)+lencds+sampOffset)<=lencds)
        %Delay matches original pulse
            ds_mod_tx(i+rowOffset,1+dso(j)+sampOffset:dso(j)+lencds+sampOffset)=...
                ref_bpsk01_code(j,:) +...
                ds_mod_tx(i+rowOffset,1+dso(j)+sampOffset:dso(j)+lencds+sampOffset);
        elseif ((1+dso(j)+sampOffset)<0 & (1+dso(j)+sampOffset)>-lencds)
        %Delay starts at previous pulse and wraps into current pulse
            ds_mod_tx(i+rowOffset,1:lencds+sampOffset+dso(j))=...
                ref_bpsk01_code(j,1-sampOffset-dso(j):lencds) +...
                ds_mod_tx(i+rowOffset,1:lencds+sampOffset+dso(j));
            if (i+rowOffset>1)
                rowOffset=rowOffset-1;
                ds_mod_tx(i+rowOffset,lencds+sampOffset+(1+dso(j)):lencds)=...
                    ref_bpsk01_code(j,1:-sampOffset-(1+dso(j))+1) +...
                    ds_mod_tx(i+rowOffset,lencds+sampOffset+(1+dso(j)):lencds);
            end
        elseif ((1+dso(j)+sampOffset+lencds)>lencds & ...
                (1+dso(j)+lencds+sampOffset)<(2*lencds))
        %Delay starts in current pulse and wraps into next pulse
            ds_mod_tx(i+rowOffset,(1+dso(j))+sampOffset:lencds)=...
                ref_bpsk01_code(j,1:lencds-sampOffset-(1+dso(j))+1) +...
                ds_mod_tx(i+rowOffset,(1+dso(j))+sampOffset:lencds);
            if (i+rowOffset<row)
                rowOffset=rowOffset+1;
                ds_mod_tx(i+rowOffset,1:dso(j)+sampOffset)=...
                    ref_bpsk01_code(j,lencds-sampOffset-(1+dso(j))+2:lencds) +...
                    ds_mod_tx(i+rowOffset,1:dso(j)+lencds-(lencds-sampOffset));
            end
        elseif (1+dso(j)+sampOffset)>lencds
        %Delay begins in next pulse and wraps to 2nd next pulse
            if (i+rowOffset<row)
                rowOffset=rowOffset+1;
                ds_mod_tx(i+rowOffset,1+dso(j)+sampOffset-lencds:lencds)=...
                    ref_bpsk01_code(j,1:2*lencds-(1+dso(j)+sampOffset)+1) +...
                    ds_mod_tx(i+rowOffset,1+dso(j)+sampOffset-lencds:lencds);
                if (i+rowOffset<row)
                    rowOffset=rowOffset+1;
                    ds_mod_tx(i+rowOffset,1:1+dso(j)+sampOffset-lencds)=...
                        ref_bpsk01_code(j,1:1+dso(j)+sampOffset-lencds) +...
                        ds_mod_tx(i+rowOffset,1:1+dso(j)+sampOffset-lencds);
                end
            end
        elseif ((1+dso(j)+lencds+sampOffset)<1)
        %Delay begins in 2nd previous pulse and wraps to previous pulse
            if (i+rowOffset>1)
                rowOffset=rowOffset-1;
                ds_mod_tx(i+rowOffset,1:lencds-(1-(1+dso(j)+sampOffset+lencds)))=...
                    ref_bpsk01_code(j,1:lencds-(1-(1+dso(j)+sampOffset+lencds))) +...
                    ds_mod_tx(i+rowOffset,1:lencds-(1-(1+dso(j)+sampOffset+lencds)));
                if (i+rowOffset>1)
                    rowOffset=rowOffset-1;
                    ds_mod_tx(i+rowOffset,lencds-(1-(1+dso(j)+...
                        sampOffset+lencds)):lencds)=...
                        ref_bpsk01_code(j,lencds-(1-(1+dso(j)+...
                        sampOffset+lencds)):lencds) +...
                        ds_mod_tx(i+rowOffset,lencds-(1-(1+dso(j)+...
                        sampOffset+lencds)):lencds);
```

```
                            end
                        end
                    end
                end
                %DEBUG MP for DS:
                %if j>1
                %if i>1
                %    subplot(4,1,1);plot(ds_mod_tx(i-1,:));
                %end
                %subplot(4,1,2);plot(ds_mod_tx(i,:));
                %disp(['mpo=',num2str(mpo(j,((i-1)*mp_num+mp_loop))),' rowoffset=',...
                %num2str(rowOffset),' sampOffset=',num2str(sampOffset),' (1+dso(j))=',...
                %num2str((1+dso(j)))]);
                %subplot(4,1,3);plot(ds_mod_tx(i+1,:));
                %subplot(4,1,4);plot(ds_mod_tx(i+2,:));pause
                %end
            end
        end
        ds_users_tx=ds_users_tx+ds_mod_tx;
    end
end%DS TX loop

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%TX: TH-PPM%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if(max(th_error_flag)==1)
    if mp_num~=0                                          %Setup Multipath values if needed:
        mpo=round(15.4e-9/dt*randn(nb_users,mp_num*row));%Match Foerster2002 results
    end
    data_source(:,2:nb_users)=randint(row,nb_users-1);data_source(:,1)=data_source1;
    if sync_flag==0
        async=0;%%%async=0: PERFECT SYNCHRONIZATION for USERS
    elseif sync_flag==1
        async=Tc;
    end
    tho(2:nb_users)=randint(1,nb_users-1,async);tho(1)=0;%th_user_offset:async. amongst diff. users
    for j=1:nb_users
        for i=1:row
            %Column values for TH offsets:%%%
            begc(j,i)=1+th_ck(j,i)+tho(j);  %
            tempbegc(j,i)=begc(j,i);        %
            if (begc(j,i)>=lencth)          %
                begc(j,i)=lencth;           %
            end                             %
            endc(j,i)=begc(j,i)-1+Tc;       %
            tempendc(j,i)=endc(j,i);        %
            if (endc(j,i)>lencth)           %
                endc(j,i)=lencth;           %
            end                           %%%
            if data_source(i,j)==0
                th_mod_tx(i,begc(j,i):endc(j,i))=th_mod_tx(i,begc(j,i):endc(j,i)) +...
                    ref_ppm0(1:(endc(j,i)-begc(j,i)+1));%ref_ppm0_code(j,1:lencth-tho(j));
            else
                th_mod_tx(i,begc(j,i):endc(j,i))=th_mod_tx(i,begc(j,i):endc(j,i)) +...
                    ref_ppm1(1:(endc(j,i)-begc(j,i)+1));%ref_ppm1_code(j,1:lencth-tho(j));
            end
            begc(j,i)=tempbegc(j,i);endc(j,i)=tempendc(j,i);
            for mp_loop=1:mp_num    %MULTIPATH added from each user/pulse
                rowOffset=fix((mpo(j,((i-1)*mp_num+mp_loop)))/lencth);
                sampOffset=round(rem(mpo(j,((i-1)*mp_num+mp_loop)),lencth));
                if (i+rowOffset>0 & i+rowOffset<=row)
                    if ((begc(j,i)+sampOffset)>0 & (endc(j,i)+sampOffset)<lencth)
                    %Delay exists within current pulse
                        th_mod_tx(i+rowOffset,begc(j,i)+sampOffset:endc(j,i)+sampOffset)=ref_ppm0+...
                            th_mod_tx(i+rowOffset,begc(j,i)+sampOffset:endc(j,i)+sampOffset);
                    elseif ((begc(j,i)+sampOffset)<0 & (begc(j,i)+sampOffset)>-Tc)
                    %Delay begins in previous pulse and wraps into current pulse
                        th_mod_tx(i+rowOffset,1:endc(j,i)+sampOffset)=...
                            ref_ppm0(1-sampOffset-begc(j,i)+1:Tc) +...
                            th_mod_tx(i+rowOffset,1:endc(j,i)+sampOffset);
                        if (i+rowOffset>1)
                            rowOffset=rowOffset-1;
                            th_mod_tx(i+rowOffset,lencth+sampOffset+begc(j,i):lencth)=...
                                ref_ppm0(1:-sampOffset-begc(j,i)+1) +...
                                th_mod_tx(i+rowOffset,lencth+sampOffset+begc(j,i):lencth);
                        end
                    elseif ((endc(j,i)+sampOffset)>lencth & (endc(j,i)+sampOffset)<(lencth+Tc))
                    %Delay begins in current pulse and wraps into next pulse
```

```
                                    th_mod_tx(i+rowOffset,begc(j,i)+sampOffset:lencth)=...
                                        ref_ppm0(1:lencth-sampOffset-begc(j,i)+1) +...
                                        th_mod_tx(i+rowOffset,begc(j,i)+sampOffset:lencth);
                                    if (i+rowOffset<row)
                                        rowOffset=rowOffset+1;
                                        th_mod_tx(i+rowOffset,1:endc(j,i)-(lencth-sampOffset))=...
                                            ref_ppm0(lencth-sampOffset-begc(j,i)+2:Tc) +...
                                            th_mod_tx(i+rowOffset,1:endc(j,i)-(lencth-sampOffset));
                                    end
                                elseif ((begc(j,i)+sampOffset)>=lencth & ((begc(j,i)+d+sampOffset)<...
                                            (lencth+fix((begc(j,i)+sampOffset)/lencth)*lencth)) )
                                    %Delay begins in next pulse
                                    if (i+rowOffset<row)
                                        rowOffset=rowOffset+1;
                                        %disp(['begc: ',num2str(begc(j,i)),'sampoff: ',num2str(sampOffset),...
                                        %'rowoff: ',num2str(rowOffset)]);
                                        th_mod_tx(i+rowOffset,begc(j,i)+sampOffset-lencth:endc(j,i)-d+...
                                            sampOffset-lencth)=ref_ppm0(1:d) +...
                                            th_mod_tx(i+rowOffset,begc(j,i)+sampOffset-lencth:...
                                            endc(j,i)-d+sampOffset-lencth);
                                    end
                                elseif ((endc(j,i)+sampOffset)<=1)
                                %Delay begins in previous pulse
                                    if (i+rowOffset>1)
                                        rowOffset=rowOffset-1;
                                        th_mod_tx(i+rowOffset,1+lencth-(1-(begc(j,i)+sampOffset)):...
                                            lencth-(1-(begc(j,i)+sampOffset))+Tc)=ref_ppm0 +...
                                            th_mod_tx(i+rowOffset,1+lencth-(1-(begc(j,i)+sampOffset)):...
                                            lencth-(1-(begc(j,i)+sampOffset))+Tc);
                                    end
                                end
                            end
                            %DEBUG MP for TH:
                            %if i>1
                            %    subplot(4,1,1);plot(th_mod_tx(i-1,:));
                            %end
                            %subplot(4,1,2);plot(th_mod_tx(i,:));
                            %disp(['mpo=',num2str(mpo(j,((i-1)*mp_num+mp_loop))),' rowoffset=',...
                            %num2str(rowOffset),' sampOffset=',num2str(sampOffset),' begc(j,i)=',...
                            %num2str(begc(j,i))]);
                            %subplot(4,1,3);plot(th_mod_tx(i+1,:));
                            %subplot(4,1,4);plot(th_mod_tx(i+2,:));pause
                        end
                    %DEBUG:
                    %if j==5
                    %    figure(1300)
                    %    plot(th_mod_tx(i,:));
                    %    i
                    %    pause
                    %end
                    end
                    th_users_tx=th_mod_tx;
                end
                %DEBUG:
                %for i=1:row
                %plot(th_users_tx(i,:));nb_users
                %pause
                %end
            end%TH TX loop

%---LOOP4
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %%%CHANNEL%%%%%%%%%%%%%%%%%%%%%%%%%
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %%%Begin loop for multiple BER trials:
            pass_ber=1;
            while(loop_ber~=0)
                if (ds_error_flag(pass_ber)==1 | th_error_flag(pass_ber)==1)
                    %if (mod(pass_pls,10)==1)
                    %Create new noise matrices every '80*row' pulses (~10,000)
                    %for speed optimization w/o perf. impact
                    %Commented because this IDEA DOES NOT work...affects performance!
                    if NBI_flag==0                       %No NBI
                        degrade1=1/10^(SNRdB(pass_ber)/10);%Reciprocal of SNR for noise
                    elseif NBI_flag~=0                   %Some form of NBI
                        degrade1=1/10^((10-10*log10(lencds/2))/10);%Force Eb/No=10dB
                    end
```

```
degrade=cov(Vt(1:Tc))*degrade1;     %Noise Power to obtain SNR
nds=sqrt(degrade)*randn(row,lencds);
nth=sqrt(degrade/lenc)*randn(row,lencth);
NBIds=0;NBIth=0;Rcj=0;              %Default: No NBI
if NBI_flag~=0
    %This section uses DS-BPSK with various codes to find performance of DS system
    %in AWGN with Narrow Band Interference (NBI) to compare results with published
    %literature [Foerster 2002].  For comparison, NBI is defined as "traditional
    %single-carrier BPSK modulated waveform."
    degrade2=1/10^(SNRdB(pass_ber)/10);%Reciprocal of SNR for Jammer
    degrade=cov(Vt(1:Tc))*degrade2;
    Jds=sqrt(degrade);
    Jth=sqrt(degrade/lenc);
    if NBI_flag==1
        %802.11b DSSS in 2.4-2.48GHz range operates at
        %11 Mbps and uses equivalent of QPSK for purposes of
        %interference (actually DQPSK with CCK modulation).
        %11-bit Barker code taken from [Skolnik].
        %Use twice desired freq. for simulation purposes
        %since inserted into demodulator (not channel).
        fcj1=2*2.412e9;   %Center freq. of jammer, channel 1
        fcj2=2*2.437e9;   %Center freq. of jammer, channel 2
        fcj3=2*2.462e9;   %Center freq. of jammer, channel 3
        Rcj=11e6;       %Chip rate of jammer=11Mbps
        Tcj=round(1/(Rcj*dt));%Nb samples in jammer chip
        %barker11=[1 1 1 0 0 0 1 0 0 1 0];
        arg1=(2*pi*fcj1*[0:dt:row*(dt*lencth)-dt]);
        arg2=(2*pi*fcj2*[0:dt:row*(dt*lencth)-dt]);
        arg3=(2*pi*fcj3*[0:dt:row*(dt*lencth)-dt]);
        po=2*pi*rand;   %Phase offset
                        %Allow QPSK:
        cos1_orig =cos(arg1+po);       cos2_orig =cos(arg2+po);       ...
            cos3_orig =cos(arg3+po);
        cos1_orig0=cos(arg1+po);       cos2_orig0=cos(arg2+po);       ...
            cos3_orig0=cos(arg3+po);
        cos1_orig1=cos(arg1+po+pi/2);  cos2_orig1=cos(arg2+po+pi/2);  ...
            cos3_orig1=cos(arg3+po+pi/2);
        cos1_orig2=cos(arg1+po+pi);    cos2_orig2=cos(arg2+po+pi);    ...
            cos3_orig2=cos(arg3+po+pi);
        cos1_orig3=cos(arg1+po+3*pi/2);cos2_orig3=cos(arg2+po+3*pi/2);...
            cos3_orig3=cos(arg3+po+3*pi/2);
        for k=1:floor(row*lencth/Tcj)
            qval=randint(1,1,4);
            cos1_orig((k-1)*Tcj+1:k*Tcj)=eval(['cos1_orig' ...
                    num2str(qval) '((k-1)*Tcj+1:k*Tcj)']);
            cos2_orig((k-1)*Tcj+1:k*Tcj)=eval(['cos2_orig' ...
                    num2str(qval) '((k-1)*Tcj+1:k*Tcj)']);
            cos3_orig((k-1)*Tcj+1:k*Tcj)=eval(['cos3_orig' ...
                    num2str(qval) '((k-1)*Tcj+1:k*Tcj)']);
        end
        cos_orig=.8165*cos1_orig+.8165*cos2_orig+.8165*cos3_orig;
        %sqrt(2/3)=.8165 factor to reduce power to expected levels
        for i=1:row
            cos_matrix(i,1:lencth)=cos_orig((i-1)*lencth+1:i*lencth);
        end
    elseif NBI_flag==2
        %Based on 802.11b DSSS in 2.4-2.48GHz range but
        %instead operates in 5GHz range using the same
        %11 Mbps and equivalent of QPSK for purposes of
        %interference (actually DQPSK with CCK modulation).
        %11-bit Barker code taken from [Skolnik]. Changed
        %the freq. to be in 10 GHz to match 2nd deriv. of UWB
        %waveform operating in 10 GHz range (not 5 GHz) since
        %direct insert in demodulator for simulation.
        fcj1=10.412e9;   %Center freq. of jammer, channel 1
        fcj2=10.437e9;   %Center freq. of jammer, channel 2
        fcj3=10.462e9;   %Center freq. of jammer, channel 3
        Rcj=11e6;       %Chip rate of jammer=11Mbps
        Tcj=round(1/(Rcj*dt));%Nb samples in jammer chip
        %barker11=[1 1 1 0 0 0 1 0 0 1 0];
        arg1=(2*pi*fcj1*[0:dt:row*(dt*lencth)-dt]);
        arg2=(2*pi*fcj2*[0:dt:row*(dt*lencth)-dt]);
        arg3=(2*pi*fcj3*[0:dt:row*(dt*lencth)-dt]);
        po=2*pi*rand;   %Phase offset
```

```
                        %Allow QPSK:
        cos1_orig =cos(arg1+po);       cos2_orig =cos(arg2+po);       ...
            cos3_orig =cos(arg3+po);
        cos1_orig0=cos(arg1+po);       cos2_orig0=cos(arg2+po);       ...
            cos3_orig0=cos(arg3+po);
        cos1_orig1=cos(arg1+po+pi/2);  cos2_orig1=cos(arg2+po+pi/2);  ...
            cos3_orig1=cos(arg3+po+pi/2);
        cos1_orig2=cos(arg1+po+pi);    cos2_orig2=cos(arg2+po+pi);    ...
            cos3_orig2=cos(arg3+po+pi);
        cos1_orig3=cos(arg1+po+3*pi/2);cos2_orig3=cos(arg2+po+3*pi/2);...
            cos3_orig3=cos(arg3+po+3*pi/2);

        for k=1:floor(row*lencth/Tcj)
            qval=randint(1,1,4);
            cos1_orig((k-1)*Tcj+1:k*Tcj)=eval(['cos1_orig' num2str(qval)...
                    '((k-1)*Tcj+1:k*Tcj)']);
            cos2_orig((k-1)*Tcj+1:k*Tcj)=eval(['cos2_orig' num2str(qval)...
                    '((k-1)*Tcj+1:k*Tcj)']);
            cos3_orig((k-1)*Tcj+1:k*Tcj)=eval(['cos3_orig' num2str(qval)...
                    '((k-1)*Tcj+1:k*Tcj)']);
        end
        cos_orig=cos1_orig+cos2_orig+cos3_orig;
        for i=1:row
            cos_matrix(i,1:lencth)=.6667*cos_orig((i-1)*lencth+1:i*lencth);
        end
    elseif NBI_flag==3
        %802.11 FHSS in 2.4-2.48GHz range operates at
        %2 Mbps and uses equivalent of QPSK for purposes of
        %interference (actually DQPSK). 79 channels each actually have
        %1 MHz of bandwidth (simulation=2MHz). FH 802.11 has 10 hops/s.
        %%
        %Instead, want to simulate FH occurring over PRI for testing so
        %use 1/(lencth*dt*row) hops/s for simulating theoretical system.
        %Use (1e6/hopf) phase_shift/s for simulating theoretical system.
        %Tcj*dt=time between phase shifts.
        %Range over 2 to 8 GHz for simulation testing.
        Rcj=1e6;                       %Chip rate of jammer=1Mbps
        hopf=1;                        %Hop factor (>=1)
        Tcj=round(hopf/(Rcj*dt));      %Nb samples in jammer chip
        %fcj1=2.4e9+1e6*randint(1,1,80);%Center freq. of jammer,channel 1
                                       %hops to new freq. each time.
        fcj1=2e9+1e7*randint(1,1,600);
        arg1=(2*pi*fcj1*[0:dt:row*(dt*lencth)-dt]);
        po=2*pi*rand;   %Phase offset
                        %Allow QPSK:
        cos1_orig =cos(arg1+po);
        cos1_orig0=cos(arg1+po);
        cos1_orig1=cos(arg1+po+pi/2);
        cos1_orig2=cos(arg1+po+pi);
        cos1_orig3=cos(arg1+po+3*pi/2);

        for k=1:floor(row*lencth/Tcj)
            qval=randint(1,1,4);
            cos1_orig((k-1)*Tcj+1:k*Tcj)=eval(['cos1_orig' num2str(qval) ...
                    '((k-1)*Tcj+1:k*Tcj)']);
        end
        cos_orig=cos1_orig;
        for i=1:row
            cos_matrix(i,1:lencth)=1.4142*cos_orig((i-1)*lencth+1:i*lencth);
            %sqrt(2) factor to get appropriate power level(SINR)
        end
    end
    NBIds=Jds*cos_matrix(:,1:lencds);
    NBIth=Jth*cos_matrix(:,1:lencth);
end%NBI_flag

%SNR CHECKS:
if (SNRcheckflag==1)
    %Perform this check 1st time through each SNR level only.
    %M  =Multiple Access Interference       SMR=Signal/MAI
    %J  =Narrow Band Interference (Jammer)  SJR=Signal/NBI
    %I  =All interference                   SIR=Signal/(N+M+J)
    signal=repmat(Vt(1:Tc),1,lenc);
    checksig=cov(signal);
    checkMds=cov(ds_users_tx(1,:))-checksig;
    checkMth=cov(th_mod_tx(1,:))-checksig;             %Remove INF check
    checkJds=cov(NBIds(1,:));                            warning off;
    checkJth=cov(NBIth(1,:));
    checkSJRds=10*log10(checksig/checkJds);
    checkSJRth=10*log10(checksig/checkJth);
    checkNds=cov(nds(1,:));
    checkNth=cov(nth(1,:));
    checkSNRds=10*log10(checksig/checkNds);
    checkSNRth=10*log10(checksig/checkNth);
    checkSMRds=10*log10(checksig/checkMds);
    checkSMRth=10*log10(checksig/checkMth);
    checkSIRds=10*log10(checksig/(checkNds+checkMds +checkJds));
    checkSIRth=10*log10(checksig/(checkNth+checkMth +checkJth));
    checkMNRds=10*log10(checkMds/checkNds);
    checkMNRth=10*log10(checkMth/checkNth);
    checkJNRds=10*log10(checkJds/checkNds);
    checkJNRth=10*log10(checkJth/checkNth);            warning on;
    Eboffsetds=10*log10((lencds)/2);
    Eboffsetth=10*log10((Tc)/2);
    checkEbNodBds(1+loop_reset-loop_ber)=checkSNRds+Eboffsetds;
    checkEbNodBth(1+loop_reset-loop_ber)=checkSNRth+Eboffsetth;
    EbNodBds(pass_ber)=SNRdB(pass_ber)+Eboffsetds;
    EbNodBth(pass_ber)=SNRdB(pass_ber)+Eboffsetth +10*log10(lenc);
    SNJRdBds(pass_ber)=10*log10(checksig/(checkNds+checkJds));
    SNJRdBth(pass_ber)=10*log10(checksig/(checkNth+checkJth));
    if save_flag==1
        fid=fopen(['data/',codetype,num2str(lenc),num2str(pass_lim),num2str(var_looplim),...
                num2str(mp_num),num2str(modul),num2str(NBI_flag),num2str(sync_flag),'.txt'],'a');
    else
        fid=1;
    end
    if modul_flag~=2
        fprintf(fid,...
            'SNRds=%5.1f SMRds=%5.1f SIRds=%5.1f MNRds=%5.1f JNRds=%5.1f EbNods=%5.1f SJRds=%5.1f\n',..
            checkSNRds,checkSMRds,checkSIRds,checkMNRds,checkJNRds,...
            EbNodBds(1+loop_reset-loop_ber),checkSJRds);
    end
    if modul_flag~=1
        fprintf(fid,...
            'SNRth=%5.1f SMRth=%5.1f SIRth=%5.1f MNRth=%5.1f JNRth=%5.1f EbNoth=%5.1f SJRds=%5.1f\n',..
            checkSNRth,checkSMRth,checkSIRth,checkMNRth,checkJNRth,...
            EbNodBth(1+loop_reset-loop_ber),checkSJRth);
    end
    if save_flag==1
        fclose(fid);
    end
end
if modul_flag~=2
    ds_channel=ds_users_tx  +nds +NBIds;
end
if modul_flag~=1
    th_channel=th_users_tx  +nth +NBIth;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%RECEIVER%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%RX: DS-BPSK%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if(ds_errors(pass_ber)<=errcutds(pass_ber))
    ds_user_rx=ds_channel;
    %Correlate input with 2 references:
    for i=1:row
        zpos=sum(ref_bpsk1_code(1,:).*ds_user_rx(i,1:lencds)); %.*dt for energy
        zneg=sum(ref_bpsk0_code(1,:).*ds_user_rx(i,1:lencds)); %.*dt for energy
        %Decision stage:
        if (zpos>=zneg) %binary 1 case:
            %ds_demod_rx(1:i+1,pass_ber)=[ds_demod_rx(1:i,pass_ber); 1];
            ds_demod_rx(i+1,pass_ber)=[1];
        else            %binary 0 case:
            ds_demod_rx(i+1,pass_ber)=[0];
        end
    end
    rowds=length(ds_demod_rx)-1;
    ds_symbols(pass_ber)=ds_symbols(pass_ber)+row;
    ds_errors(pass_ber)=ds_errors(pass_ber)+...
        length(find(xor(ds_demod_rx(2:rowds+1,pass_ber),data_source1)));
end%DS RX error loop
```

```
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%%RX: TH-PPM%%%%%%%%%%%%%%%%%%%%%%
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        if(th_errors(pass_ber)<=errcutth(pass_ber))
            th_user_rx=th_channel;
            %Correlate input with reference:
            for i=1:row
                zppm0=sum(ref_ppm0.*th_user_rx(i,begc(1,i):endc(1,i))); %.*dt for energy
                zppm1=sum(ref_ppm1.*th_user_rx(i,begc(1,i):endc(1,i))); %.*dt for energy
                %Decision stage:
                if (zppm1>=zppm0)   %Binary 1 case:
                    %th_demod_rx(1:i+1,pass_ber)=[th_demod_rx(1:i,pass_ber); 1];
                    th_demod_rx(i+1,pass_ber)=[1];
                else                %Binary 0 case:
                    %th_demod_rx(1:i+1,pass_ber)=[th_demod_rx(1:i,pass_ber); 0];
                    th_demod_rx(i+1,pass_ber)=[0];
                end
            end
            rowth=length(th_demod_rx)-1;
            th_symbols(pass_ber)=th_symbols(pass_ber)+row;
            th_errors(pass_ber)=th_errors(pass_ber)+...
                length(find(xor(th_demod_rx(2:rowth+1,pass_ber),data_source1)));
        end%TH RX error loop

        %fprintf('Completed BER Pass: %d \n',pass_ber);
        if (th_errors(pass_ber)>errcutth(pass_ber) & th_repeat(pass_ber)==1)
            %Following 'if' statement changes the error cutoff threshold
            %for TH case (same for DS below) to the appropriate level
            %based on the expected BER. See Raj Jain text for eqn. to
            %determine appropriate threshold settings. Min. threshold set
            %at 11 to maintain np>10 requirement (see Misc. Info below).
            if (th_errors(pass_ber)/th_symbols(pass_ber)) <1e-4
                errcutth(pass_ber)=300; %=11 for r=+/- 59%
            elseif (th_errors(pass_ber)/th_symbols(pass_ber)) <1e-2
                errcutth(pass_ber)=300; %=30 for r=+/- 35%
            elseif (th_errors(pass_ber)/th_symbols(pass_ber)) <1
                errcutth(pass_ber)=300; %r=+/- 11%
            end
            if (th_errors(pass_ber)>errcutth(pass_ber) & th_repeat(pass_ber)==1)
                th_error_flag(pass_ber)=0;
                th_repeat(pass_ber)=0;
                %fprintf('  Exceed %u TH errors for SNR level %d\n',errcutth(pass_ber),...
                %SNRdB(pass_ber));
            end
        end
        if (ds_errors(pass_ber)>errcutds(pass_ber) & ds_repeat(pass_ber)==1)
            if (ds_errors(pass_ber)/ds_symbols(pass_ber)) <1e-4
                errcutds(pass_ber)=300; %=11 for r=+/- 59%
            elseif (ds_errors(pass_ber)/ds_symbols(pass_ber)) <1e-2
                errcutds(pass_ber)=300;  %=30 for r=+/- 35%
            elseif (ds_errors(pass_ber)/ds_symbols(pass_ber)) <1
                errcutds(pass_ber)=300; %r=+/- 11%
            end
            if (ds_errors(pass_ber)>errcutds(pass_ber) & ds_repeat(pass_ber)==1)
                ds_error_flag(pass_ber)=0;
                ds_repeat(pass_ber)=0;
                %fprintf('  Exceed %u DS errors for SNR level %d\n',errcutds(pass_ber),...
                %SNRdB(pass_ber));
            end
        end
    end%end error_flag if statement after channel
    nberrors=min([min(th_errors(:)) min(ds_errors(:))]);
    pass_ber=pass_ber+1;
    loop_ber=loop_ber-1;
    if nargin==0
        disp('No arguments passed to function.  ');
        disp('Completed inner-most loop.        ');
        disp('Stopping for Signal Generation.   ');
        plotdata(Vt,row,dt,Twp,T,To,Tc,ref_matrix,NBIth,Rcj);
        close(mywaitbar);
        return
    end
end%%%Multiple BER Trials:  LOOP4

if (mod(pass_pls,10000)==1)
    fprintf('Completed Pulse Pass: %d for %d users\n',pass_pls,nb_users);
```

```
        if modul_flag~=2
            fprintf('Number of errors DS: ');fprintf('%d ',ds_errors);fprintf(' Pb=%.1e ',...
                ds_errors/(pass_pls*row));fprintf('\n');
        end
        if modul_flag~=1
            fprintf('Number of errors TH: ');fprintf('%d ',th_errors);fprintf(' Pb=%.1e ',...
                th_errors/(pass_pls*row));fprintf('\n');
        end
    end
    SNRcheckflag=0;
    pass_pls=pass_pls+1;
    loop_ber=loop_reset;
end%%%Multiple Pulse Matrices Trials:   LOOP3

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%ANALYSIS%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if show_analysis==1
    if save_flag==1
        fid=fopen(['data/',codetype,num2str(lenc),num2str(pass_lim),num2str(var_looplim),...
                num2str(mp_num),num2str(modul),num2str(NBI_flag),num2str(sync_flag),'.txt'],'a');
    else
        fid=1;
    end
    fprintf(fid,'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&\n');
    fprintf(fid,'VarPass=%d\tNbusers=%d\t%s%d\n',var_loop,nb_users,codetype,lenc);
    fprintf(fid,'\t\t\t\t BER Performance \n');
    fprintf(fid,'\tBER:  \t\tNb_Sym_Err \t\tNb_Sym: \tSNR \tEbNodB \tErrCut:\n');
    for i=1:(pass_ber-1)
        if modul_flag~=2
            ds_ber(i)=ds_errors(i)/ds_symbols(i);
            fprintf(fid,'DS:\t%f %8d %14d %10.1f %8.2f %5d\n',ds_ber(i),ds_errors(i),...
                ds_symbols(i),SNRdB(i),EbNodBds(i),errcutds(i));
        end
        if modul_flag~=1
            th_ber(i)=th_errors(i)/th_symbols(i);
            fprintf(fid,'TH:\t%f %8d %14d %10.1f %8.2f %5d\n',th_ber(i),th_errors(i),...
                th_symbols(i),SNRdB(i)+10*log10((lenc)),EbNodBth(i),errcutth(i));
        end
    end
    if (length(SNRdB)>1)
        if NBI_flag==0
            %Generate BER Curves:
            figure(400+nb_users);
            semilogy(EbNodBds,ds_ber,'*');
            xlabel('E_b/N_o (dB)');ylabel('P_b');
            title(['Nb.Users=',num2str(nb_users),' MP:',num2str(mp_num),' Of:',num2str(async),...
                    'NBI:',num2str(NBI_flag),codetype,num2str(lenc)]);
            hold on;
            semilogy(EbNodBth,th_ber,'x');
            if nb_users<2
                %Analytic Expressions of Q(x):
                EbNods=10.^(.1*EbNodBds);
                EbNoth=10.^(.1*EbNodBth);
                Qantipodal=.5*erfc(sqrt(2*EbNods)/sqrt(2));
                Qorthogonal=.5*erfc(sqrt(EbNoth)/sqrt(2));
                if var_loop==1
                    semilogy(EbNodBds,Qantipodal);
                    semilogy(EbNodBth,Qorthogonal,'--');
                end
                legend('DS-BPSK(Antipodal)','TH-PPM(Orthogonal)','Analytic (Antipodal)',...
                    'Analytic (Orthogonal)');
                ds_abserror=sum((ds_ber-Qantipodal).^2);
                ds_variance=1/(length(ds_ber)-1)*sum((ds_ber-Qantipodal).^2);
                th_abserror=sum((ds_ber-Qantipodal).^2);
                th_variance=1/(length(th_ber)-1)*sum((th_ber-Qorthogonal).^2);
                fprintf('DS: Abs. Error: %e\t Variance: %e\n',ds_abserror,ds_variance);
                fprintf('TH: Abs. Error: %e\t Variance: %e\n',th_abserror,th_variance);
            end
            %hold off;
        elseif NBI_flag~=0
            figure(450+nb_users);
            if modul_flag~=2
                semilogy(SNJRdBds,ds_ber,'*');
            end
            if modul_flag~=1
                semilogy(SNJRdBth,th_ber,'x');
            end
            xlabel('SNJR (dB)');ylabel('P_b');
```

```
                    title(['Eb/No=10 (SNR:' num2str(10-10*log10(lencds/2)) 'dB)  Nb. Users= ',...
                            num2str(nb_users),'  ',codetype,num2str(lenc),'    NBFreq= ', num2str(fcj1)]);
                    hold on;
                    legend('DS-BPSK(Antipodal)','TH-PPM(Orthogonal)');
                end%NBI_flag for plotting
                if save_flag==1
                    saveas(gcf,['results/',codetype,num2str(lenc),num2str(nb_users),num2str(var_looplim),...
                            num2str(mp_num),num2str(modul),num2str(NBI_flag),num2str(sync_flag),'msnr'],'fig');
                end
            end
            fprintf(fid,'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&\n');
            if save_flag==1
                fclose(fid);
            end
            ds_ber_user(var_loop,pass_user)=ds_ber(length(ds_ber));
            th_ber_user(var_loop,pass_user)=th_ber(length(th_ber));
            EbNodBvards(var_loop,pass_user)=EbNodBds(length(SNRdB));
            EbNodBvarth(var_loop,pass_user)=EbNodBth(length(SNRdB));
        end%show analysis
        close(mywaitbar);
    end%pass_user:  LOOP2
    if save_flag==1
        save(['data/',codetype,num2str(lenc),num2str(nb_users),num2str(var_looplim),num2str(mp_num),...
                num2str(modul),num2str(NBI_flag),num2str(sync_flag),'.mat'],...
            'th_ber','ds_ber','SNRdB','EbNodBds','EbNodBth','ds_ber_user','th_ber_user','codetype',...
            'lenc','errcutds','errcutth','nb_users','nb_usersbeg','var_loop','var_looplim','mp_num',...
            'modul','NBI_flag','sync_flag');
    end
    if (nb_users>1)
        figure(500)
        if (var_looplim>1)
            subplot(2,1,1)
        end
        semilogy(1:nb_users,ds_ber_user(var_loop,:),'*');hold on;
        semilogy(1:nb_users,th_ber_user(var_loop,:),'x');%hold off;
        title(['Mean Eb/No= ',num2str(mean(mean(EbNodBvards(find(EbNodBvards))))),'DS,',...
                num2str(mean(mean(EbNodBvarth(find(EbNodBvards))))),'TH; MP:',num2str(mp_num),...
                ' Of:',num2str(async),' NBI:',num2str(NBI_flag),codetype,num2str(lenc)]);
        xlabel('Number of Users');ylabel('P_b');
        legend('DS-BPSK(Antipodal)','TH-PPM(Orthogonal)');
        if save_flag==1
            saveas(gcf,['results/',codetype,num2str(lenc),num2str(nb_users),num2str(var_looplim),...
                    num2str(mp_num),num2str(modul),num2str(NBI_flag),num2str(sync_flag),'musr'],'fig');
        end
    end
end%Variance:   LOOP1
if (var_looplim>1 & nb_users>1)
    figure(500)
    subplot(2,1,2)
    semilogy(1:nb_users,median(ds_ber_user),'*');hold on;
    semilogy(1:nb_users,median(th_ber_user),'x');
    errorbar(median(ds_ber_user),std(ds_ber_user));
    errorbar(median(th_ber_user),std(th_ber_user));
    title(['Median Pb: ',num2str(var_looplim),' Trials, Mean Eb/No= ',...
            num2str(mean(mean(EbNodBvards(find(EbNodBvards))))),'DS ',...
            num2str(mean(mean(EbNodBvarth(find(EbNodBvards))))),'TH; MP:',...
            num2str(mp_num),' Of:',num2str(async),' NBI:',num2str(NBI_flag),codetype,num2str(lenc)]);
    xlabel('Number of Users');ylabel('P_b');
    legend('DS-BPSK(Antipodal)','TH-PPM(Orthogonal)');
    if save_flag==1
        saveas(gcf,['results/',codetype,num2str(lenc),num2str(nb_users),...
                num2str(var_looplim),num2str(mp_num),num2str(modul),num2str(NBI_flag),...
                num2str(sync_flag),'musr'],'fig');
    end
end
tstop=etime(clock,tstart); disp(['Execution time:
',num2str(tstop),' seconds or ',num2str(tstop/3600), ' hours.']);

%MISC. INFO:
%--This outputs complementary error function Q(x):
%   x=.33;Q=.5*erfc(x/sqrt(2))
%--For mean & STD calculation on one line:
%   x=10.^(.1*[-15:2:5]);Qantipodal=.5*erfc(sqrt(2*x)/sqrt(2));Qorthogonal=.5*erfc(sqrt(x)/sqrt(2));...
%     fprintf('Qantipodal \n');fprintf('%f\n',Qantipodal);fprintf('Qorthogonal \n');fprintf('%f\n',Qorthogonal)
%--Conversion from SNR to Eb/No:
%   Eb/No=SNR +10*log10(Ts/(2*k*dt));
%     Ts=length of samples into correlator
%     k =bits/symbol=1 for binary case (Mary=2^k)
%     dt=sampling resolution
```

B.2 codeselect.m

```
function[code,codetype,lenc]=codeselect(codeselector);
%[code,codetype,lenc]=codeselect(codeselector)
%
%codeselector=1:    No code                 , length   1
%codeselector=2:    Gold Code               , length  31
%codeselector=3:    Simulated Annealing-AC  , length  31
%codeselector=4:    Simulated Annealing-A   , length  31
%codeselector=5:    Simulated Annealing-C   , length  31
%codeselector=6:
%codeselector=7:    Mseq                    , length  31
%codeselector=8:    Random Integers         , length  31
%codeselector=9:    Gold Code               , length 127
%codeselector=12:   Gold Code               , length 511
%codeselector=13:   MPS  Code               , length  69
%codeselector=14:   Simulated Annealing-AC  , length  69
%
%Return code,type, and length of code
%Returned codes are 0,1 biphase (binary).

%codeselect.m
%Created by 2LT Courtney Canadeo
%Last modified Dec. 2002

if codeselector==1      %Default: no coding
    codetype='NoCode';
    code(1:30,1)=1;

elseif codeselector==2  %GOLD CODE: length 31
    codetype='Gold';
    gold_31bit=[1 1 1 1 0 1 1 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 0 0 1;
        1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0;
        1 1 1 0 1 1 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 0 0 1 1;
        1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0;];
    gold_31bit=gold_gen(gold_31bit(1,:),gold_31bit(2,:));
    code=gold_31bit(5:33,:);%Avoid using mseq by starting at 5(skipping 1,2)

elseif codeselector==3  %SIMULATED ANNEALING
    %SA31_15_1.00_1.00.txt
    %Wednesday, 11/6/2002, 12:32
    %N: 31, M: 15,  Number of Phases: 2
    %Code Type: Binary
    %Initial Temperature: 310.00 , Final Temperature: 0.03
    %Temp decrement: T' = 0.10T
    %Equilibrium: #accept or #reject > NM(1+0.1NM/T)
    %Cost Function: Max Sidelobes; Auto Weight: 1.00, Cross Weight: 1.00
    %Emin: 20.000000
    codetype='SAAC';
    code=[1 1 -1 -1 1 1 1 1 1 1 -1 1 1 1 -1 -1 -1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 1 -1 1
        1 -1 -1 1 -1 1 -1 1 1 -1 -1 1 -1 -1 -1 1 -1 -1 -1 1 -1 -1 -1 -1 1 1 1 -1 1 -1 1
        1 1 -1 1 -1 1 -1 -1 1 1 -1 1 -1 -1 1 -1 -1 -1 -1 1 1 1 -1 1 1 -1 1 -1 1 1 1
        1 -1 1 -1 -1 -1 1 -1 -1 -1 -1 1 -1 1 -1 1 1 -1 1 1 -1 1 1 -1 1 -1 1 1 1 1 1
       -1 -1 1 1 -1 -1 -1 1 1 -1 1 1 1 -1 1 -1 1 -1 -1 1 -1 1 1 1 1 1 -1 1 -1 1 1
        1 -1 1 1 1 1 -1 1 -1 1 -1 -1 -1 1 1 -1 -1 1 -1 1 1 1 -1 -1 1 1 1 1 1 1 1
       -1 -1 1 1 1 -1 1 -1 1 1 1 1 1 -1 -1 -1 1 -1 -1 1 -1 1 1 -1 -1 1 -1 -1 1 -1 -1
       -1 1 -1 -1 -1 -1 -1 -1 1 -1 1 -1 -1 -1 1 -1 -1 -1 1 -1 -1 -1 -1 1 1 -1 1 -1 -1 1 -1
        1 -1 -1 -1 -1 -1 -1 1 -1 -1 -1 -1 -1 1 -1 1 1 1 1 1 -1 1 1 -1 1 1 -1 -1 -1 1 1
       -1 1 -1 -1 -1 1 -1 1 1 1 1 1 1 -1 -1 -1 -1 1 -1 -1 -1 -1 -1 1 1 -1 -1 -1 1 -1 1
       -1 1 1 -1 1 1 1 1 -1 -1 -1 -1 -1 -1 1 1 -1 -1 -1 1 -1 1 -1 1 1 1 1 -1 1 -1 1
       -1 1 1 -1 1 -1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 -1 1 1 1 -1 1 -1 -1 -1 1 -1 1 1 -1
       -1 -1 1 1 -1 1 1 -1 1 -1 -1 -1 1 -1 -1 -1 -1 -1 -1 1 -1 -1 -1 -1 1 -1 -1 -1 1 1 -1
        1 -1 -1 1 -1 -1 1 -1 1 -1 -1 -1 1 1 -1 1 -1 -1 -1 1 -1 -1 -1 -1 1 1 -1 1 -1 1 1
        1 1 -1 1 -1 -1 -1 1 -1 -1 1 1 1 -1 -1 -1 -1 1 -1 -1 1 1 -1 1 -1 1 1 1 -1 1 -1];
    code(find(code==-1))=0;
    %load('SA_31bit.mat');   %SA_31bit=code for Simulated Annealing 31 bit length
    %load('SA_69bit.mat');   %SA_69bit=code for Simulated Annealing 69 bit length

elseif codeselector==4  %SIMULATED ANNEALING
    %SA_15_100_0.txt
    %Thursday, 11/7/2002, 15:54
    %N: 31, M: 15,  Number of Phases: 2
    %Code Type: Binary
    %Initial Temperature: 310.00 , Final Temperature: 0.03
    %Temp decrement: T' = 0.10T
    %Equilibrium: #accept or #reject > NM(1+0.1NM/T)
    %Cost Function: Max Sidelobes; Auto Weight: 100.00, Cross Weight: 0.00
    %Emin: 700.000000
    codetype='SAA';
    code=[-1 -1 -1 1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 -1 -1 -1 1 -1 -1 1 -1 1 -1 -1 1 -1 -1 1 -1
        -1 1 -1 -1 -1 1 -1 1 1 1 -1 -1 1 -1 1 -1 1 -1 -1 -1 1 1 -1 1 1 -1 1 -1 -1 -1 -1
        1 1 1 -1 1 1 1 1 -1 -1 -1 1 1 -1 1 1 -1 -1 -1 1 1 -1 1 1 -1 1 1 -1 1 -1 -1
        -1 1 1 -1 -1 -1 1 -1 1 -1 1 -1 -1 1 -1 -1 1 1 1 1 1 1 -1 -1 -1 -1 1 1 -1 -1 1
        -1 -1 -1 -1 1 -1 -1 -1 -1 -1 1 -1 -1 -1 1 -1 -1 -1 -1 1 1 1 1 1 1 -1 1 -1 -1 1 -1
        -1 -1 1 -1 1 1 1 -1 1 1 1 1 -1 1 1 -1 1 1 1 1 -1 -1 1 -1 1 -1 -1 -1 -1 1 1
        1 -1 1 1 -1 1 1 -1 1 1 1 1 -1 -1 -1 1 -1 1 -1 -1 1 1 -1 1 1 1 -1 1 1 1 -1
        1 -1 -1 -1 -1 -1 1 -1 1 -1 1 1 -1 -1 1 1 1 1 -1 1 1 1 -1 1 1 -1 1 -1 -1 1 -1
        1 1 1 1 -1 -1 -1 -1 -1 1 -1 1 -1 1 1 1 -1 -1 1 1 1 1 -1 1 1 1 1 1 1 -1 1
        1 -1 1 -1 -1 1 1 -1 1 -1 1 1 1 -1 -1 1 -1 1 1 1 -1 -1 -1 -1 1 1 1 1 1 -1 1
        -1 1 1 1 -1 1 1 -1 -1 1 1 -1 -1 -1 1 1 -1 -1 -1 -1 1 1 -1 -1 -1 -1 -1 1 -1 1 -1
        1 1 1 -1 1 1 -1 1 -1 -1 -1 1 -1 1 1 1 -1 1 1 1 1 -1 1 1 1 1 -1 1 -1 1 1
        -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 1 1 -1 1 -1 1 1 -1 1 1 1 -1 1 -1 -1 -1 1 1 1
        1 -1 -1 -1 1 -1 -1 -1 -1 -1 1 -1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 -1 1 -1 1 1
        -1 1 1 -1 -1 -1 1 -1 -1 1 -1 1 -1 1 -1 1 -1 -1 1 1 1 1 -1 1 -1 -1 1 1 -1 -1 -1];
    code(find(code==-1))=0;

elseif codeselector==5  %SIMULATED ANNEALING
    %SA_15_0_100.txt
    %Thursday, 11/7/2002, 17:12
    %N: 31, M: 15,  Number of Phases: 2
    %Code Type: Binary
    %Initial Temperature: 310.00 , Final Temperature: 0.03
    %Temp decrement: T' = 0.10T
    %Equilibrium: #accept or #reject > NM(1+0.1NM/T)
    %Cost Function: Max Sidelobes; Auto Weight: 0.00, Cross Weight: 100.00
    %Emin: 1300.000000
    codetype='SAC';
    code=[-1 -1 -1 -1 1 1 1 -1 1 -1 1 1 1 -1 -1 -1 1 1 1 1 1 -1 -1 1 1 1 1 1 1 -1 1
        -1 -1 -1 1 1 1 1 -1 1 1 1 -1 -1 -1 -1 -1 -1 1 1 1 -1 1 1 1 1 -1 1 1 1 1 -1
        -1 1 1 -1 1 1 -1 -1 -1 1 -1 -1 1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 1 -1 1 1 -1
        -1 -1 -1 1 1 -1 -1 1 1 -1 1 1 -1 1 1 1 -1 1 1 -1 -1 1 1 1 1 -1 -1 1 -1 -1 -1
        -1 -1 -1 1 -1 1 -1 1 -1 -1 -1 -1 1 1 -1 -1 1 -1 1 -1 -1 -1 -1 -1 -1 1 1 -1 1 -1 -1
        1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 -1 -1 1 -1 1 -1 1 -1 1 1 1 -1 -1 1 1 -1 1 -1
        -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 -1 1 -1 1 1 1 1 1 1 1 1 1 -1 -1 1 1 -1 1 -1
        -1 -1 -1 -1 -1 1 -1 -1 -1 -1 -1 1 1 1 1 -1 1 1 -1 -1 1 1 1 -1 -1 -1 -1 -1 -1 1 -1
        -1 1 -1 -1 -1 -1 1 -1 -1 1 -1 -1 -1 1 1 -1 1 1 1 1 1 -1 -1 -1 -1 1 1 -1 -1 1 -1
        1 1 1 -1 1 -1 1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 1 -1 1 -1 1 1 -1 1 1 -1 -1 -1
        -1 -1 -1 -1 1 -1 1 -1 1 -1 1 1 -1 -1 1 -1 -1 -1 -1 1 1 1 -1 1 -1 -1 -1 1 -1 -1 -1
        -1 1 1 1 1 -1 -1 1 1 1 -1 -1 -1 1 1 -1 -1 -1 -1 1 -1 -1 1 -1 1 1 -1 1 1 1 -1
        -1 -1 -1 1 -1 1 1 -1 -1 1 1 1 1 -1 1 1 1 -1 -1 1 1 1 -1 1 -1 1 -1 -1 1 1 -1
        1 1 -1 -1 -1 1 1 -1 -1 -1 1 -1 -1 1 1 1 -1 1 -1 -1 1 -1 -1 -1 1 1 -1 1 1 1 1
        1 -1 1 1 1 1 1 1 -1 -1 1 1 1 1 1 -1 -1 -1 1 -1 -1 1 1 1 -1 -1 -1 -1 -1 -1 -1   ];
    code(find(code==-1))=0;

elseif codeselector==6  %SIMULATED ANNEALING
    %SA_15_01_1000000.txt
    %Wednesday, 11/20/2002, 17:24
    %N: 31, M: 15,  Number of Phases: 2
    %Code Type: Binary
    %Initial Temperature: 310.00 , Final Temperature: 0.03
    %Temp decrement: T' = 0.10T
    %Equilibrium: #accept or #reject > NM(1+0.1NM/T)
    %Cost Function: Max Sidelobes; Auto Weight: 0.01, Cross Weight: 1000000.00
    %Emin: 13000000.090000
    codetype='SAC6';
    code=[1 1 -1 1 1 -1 1 1 -1 1 -1 1 1 1 1 1 -1 -1 -1 1 1 1 1 -1 1 1 1 -1 -1 1 1
        -1 1 -1 -1 -1 1 1 1 -1 1 1 1 1 1 -1 1 1 1 1 1 -1 -1 1 1 -1 -1 1 -1 -1 1 1
        -1 1 1 1 -1 -1 1 -1 -1 1 -1 1 -1 1 1 1 -1 1 1 -1 -1 -1 1 -1 -1 1 -1 -1 -1 -1 1
        -1 1 -1 1 1 -1 1 -1 1 -1 -1 -1 1 -1 1 -1 -1 -1 -1 -1 -1 1 1 -1 -1 1 -1 -1 -1 1 1
        1 1 -1 1 1 -1 -1 1 1 1 -1 1 -1 1 -1 -1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 1 1 -1 -1
        1 1 1 1 1 -1 1 -1 -1 1 1 -1 1 -1 1 -1 1 -1 -1 1 1 1 -1 1 1 -1 1 1 1 -1 1
        -1 1 1 -1 1 -1 -1 -1 1 -1 1 1 -1 1 -1 -1 -1 -1 -1 -1 -1 1 -1 1 -1 -1 1 1 -1 -1 -1
        -1 1 1 1 -1 1 -1 1 -1 1 -1 1 1 1 -1 -1 1 -1 -1 1 1 1 1 -1 -1 -1 1 -1 -1 1 -1
        -1 1 -1 -1 1 -1 -1 1 1 -1 -1 1 -1 -1 -1 -1 -1 1 1 1 -1 1 1 1 1 -1 1 -1 -1 -1 1
        1 1 1 1 1 -1 -1 -1 -1 1 -1 -1 -1 1 -1 -1 -1 1 1 -1 1 -1 1 -1 1 1 -1 -1 1 1 1
        -1 1 -1 1 1 1 -1 1 1 1 -1 -1 1 1 -1 1 1 -1 -1 -1 -1 1 -1 -1 1 1 -1 -1 1 -1 -1
        -1 -1 1 -1 1 -1 -1 -1 -1 1 -1 1 1 1 -1 -1 -1 -1 1 1 1 -1 1 -1 1 -1 1 1 -1 -1 1
        1 -1 -1 -1 1 1 1 -1 1 1 1 -1 -1 1 1 1 1 -1 -1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1
        -1 1 1 -1 1 1 -1 1 -1 1 1 -1 -1 1 1 1 1 1 -1 1 -1 1 -1 -1 -1 -1 -1 1 1 -1 1
        -1 1 1 1 1 -1 1 -1 1 1 1 1 1 -1 -1 1 1 1 -1 -1 1 1 1 1 -1 1 -1 -1 -1 1 -1];
    code(find(code==-1))=0;

elseif codeselector==7
    codetype='Testingmseq';
    mseq1=mSeqGen([0 1 1 1 1],[1 1 0 1 0 1 ]);
    mseq2=mSeqGen([0 1 1 0 1],[1 1 0 1 0 1 ]);
    mseq3=mSeqGen([0 1 0 1 1],[1 1 0 1 0 1 ]);
    mseq4=mSeqGen([1 0 1 0 1],[1 1 0 1 0 1 ]);
    mseq5=mSeqGen([0 1 0 0 1],[1 1 0 1 0 1 ]);
    code=[mseq1; mseq2; mseq3; mseq4 ; mseq5];

elseif codeselector==8
    codetype='Randints';
    %code=randint(20,31,2);
    code=[0  1  0  1  1  1  1  0  0  0  1  0  1  1  0  0  0  1  0  0  1  1  1  0  1  1  1  0  1  0  1
        0  1  0  1  1  1  1  0  1  1  1  1  0  1  1  1  0  0  0  0  1  0  1  1  0  0  0  0  0  1  1
        0  1  0  1  1  1  1  0  0  1  0  0  1  0  1  1  1  1  0  0  0  1  0  0  0  0  1  1  0  1  1
        0  1  0  0  1  1  0  0  0  0  0  0  1  1  0  1  0  1  0  0  0  0  0  0  0  0  0  1  1  1  0
        0  1  0  0  1  0  1  0  0  1  0  1  1  0  1  0  1  0  0  1  0  0  0  0  1  1  1  0  1  1  1
        1  0  1  1  1  0  0  0  1  1  1  0  1  0  0  0  1  1  1  0  0  0  1  0  0  0  1  0  0  0  1
        0  1  0  0  0  1  0  1  0  0  1  0  1  0  0  0  1  0  1  0  0  0  0  1  1  1  0  0  0  0  0
        0  0  0  0  1  0  1  0  1  1  1  1  0  0  0  0  1  1  0  0  0  0  1  1  1  1  1  1  0  0  1
        1  1  0  0  0  0  1  0  0  1  1  1  1  0  0  0  0  0  1  1  1  1  0  1  1  0  1  0  0  1  0
        1  1  0  0  0  0  1  1  0  1  0  0  0  1  0  0  0  0  1  1  0  0  0  1  0  0  1  0  1  0  1
        0  1  1  1  1  1  1  0  1  0  0  0  1  0  0  1  0  1  1  1  0  0  1  1  0  0  0  1  1  1  0
        0  0  0  1  1  1  1  1  1  1  1  0  0  1  0  0  1  1  1  1  0  1  1  0  0  1  1  0  0  0  1
        0  0  1  0  0  0  1  1  0  1  1  0  1  0  1  0  1  0  1  0  1  1  0  1  0  0  1  0  1  1  0
        1  0  0  1  1  1  1  1  1  0  1  1  1  1  0  0  1  1  1  1  0  0  0  0  0  0  0  1  0  1  0
        1  1  0  0  0  1  1  0  1  1  1  0  0  1  1  1  1  0  0  0  1  0  1  1  0  1  1  0  0  0  0
        0  0  1  1  1  1  1  0  0  0  0  1  0  0  0  1  1  1  1  1  0  0  0  1  0  0  1  0  0  1  0
        1  0  0  1  0  1  1  0  1  0  0  0  0  0  0  1  1  0  0  0  0  1  0  0  1  1  0  1  0  1  0
        0  1  1  1  0  0  1  0  1  0  1  0  1  0  0  0  1  0  0  1  0  1  0  0  1  0  0  1  0  1  1
        1  1  1  1  0  0  0  1  1  0  0  1  1  0  1  0  1  1  1  1  1  1  1  0  1  0  1  1  1  0  1
        0  0  0  1  1  1  1  1  1  0  0  1  0  0  1  0  1  0  1  0  1  1  1  1  0  1  0  1  0  1  1];
elseif codeselector==9
    codetype='Gold';
    mseq1=mSeqGen([1 1 1 0 1 1 1],oct2bin([3 2 3]));
    mseq2=mSeqGen([1 1 0 1 1 1 1],oct2bin([2 4 7]));
    code=gold_gen(mseq1,mseq2);
elseif codeselector==10  %SIMULATED ANNEALING
    %batch1.txt
    %Thursday, 11/21/2002, 15:33
    %N: 31, M: 15,  Number of Phases: 2
    %Code Type: Binary
    %Initial Temperature: 310.00 , Final Temperature: 0.03
    %Temp decrement: T' = 0.10T
    %Equilibrium: #accept or #reject > NM(1+0.1NM/T)
    %Cost Function: Total Sidelobes; Auto Weight: 1000000000.00, Cross Weight: 0.01
    %Emin: 7000000000.140000
    codetype='SAA10';
    code=[1 1 -1 1 -1 1 1 -1 1 -1 1 1 1 1 1 1 -1 -1 1 1 -1 -1 1 -1 -1 -1 -1 -1 -1 1 1
        1 1 -1 -1 1 1 -1 1 -1 1 -1 1 -1 -1 -1 -1 1 -1 -1 -1 1 1 1 -1 -1 1 -1 -1 1 -1 -1
        -1 1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 -1 1 1 1 1 -1 1 1 -1 -1 -1 1 -1 1 -1 -1 -1 1
        -1 1 1 1 1 -1 1 1 1 1 -1 -1 1 1 -1 1 -1 -1 1 -1 1 1 1 -1 -1 -1 1 -1 1 1 1
        -1 -1 -1 -1 1 1 -1 -1 1 -1 1 1 1 1 1 -1 -1 -1 1 -1 1 -1 -1 1 1 1 -1 1 -1 1 -1
        1 -1 1 1 -1 -1 -1 -1 1 1 -1 1 -1 1 1 1 1 1 1 1 1 -1 -1 -1 1 -1 1 1 -1 1 -1
        -1 1 1 -1 1 1 -1 1 -1 1 1 1 1 1 1 -1 1 -1 -1 1 1 -1 1 -1 -1 -1 1 1 -1 -1 -1
        1 -1 -1 1 -1 -1 -1 1 -1 -1 1 1 -1 -1 -1 1 1 1 -1 -1 -1 -1 1 -1 1 1 1 -1 1 -1 -1
        1 -1 1 -1 1 1 -1 -1 1 -1 -1 -1 -1 -1 1 1 1 1 -1 1 1 1 1 1 -1 -1 1 1 1 1 1
        -1 1 1 1 1 -1 -1 1 -1 1 -1 -1 1 1 1 -1 1 -1 -1 1 1 1 -1 1 1 1 1 1 -1 1 -1
        -1 1 1 1 1 -1 1 1 1 -1 -1 1 -1 -1 -1 1 -1 1 -1 -1 -1 1 1 1 1 -1 -1 1 -1 -1 -1
        1 1 1 1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 1 -1 1 1 -1 1 1 1 -1 1 -1 -1 -1 1 1
        -1 -1 -1 -1 -1 1 1 1 1 -1 1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 -1 -1 1 -1 1 -1 -1 -1
        1 -1 1 1 1 1 -1 1 -1 1 -1 1 1 1 1 -1 -1 1 -1 -1 1 -1 1 1 1 1 -1 1 -1 -1 1
        -1 -1 1 1 1 -1 1 -1 1 -1 1 1 -1 1 1 1 1 -1 1 -1 -1 1 -1 -1 1 1 -1 -1 -1 1 1];
    code(find(code==-1))=0;
elseif codeselector==11  %SIMULATED ANNEALING
    %batch0.txt
    %Thursday, 11/21/2002, 14:36
    %N: 31, M: 15,  Number of Phases: 2
    %Code Type: Binary
    %Initial Temperature: 310.00 , Final Temperature: 0.03
    %Temp decrement: T' = 0.10T
    %Equilibrium: #accept or #reject > NM(1+0.1NM/T)
    %Cost Function: Total Sidelobes; Auto Weight: 0.01, Cross Weight: 1000000.00
    %Emin: 13000000.090000
    codetype='SAC11';
    code=[1 -1 -1 -1 -1 -1 -1 1 1 -1 -1 1 -1 -1 1 -1 1 1 -1 1 1 1 1 1 -1 1 -1 1 -1 -1 1
        1 -1 -1 1 1 1 -1 1 -1 -1 1 -1 -1 1 1 -1 -1 -1 1 1 -1 -1 -1 -1 -1 1 -1 -1 1 -1 -1
        -1 1 1 1 1 -1 1 1 -1 1 -1 -1 -1 -1 1 1 -1 -1 1 1 1 1 1 1 1 -1 -1 1 1 -1 1
        1 -1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1 -1 1 1 -1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 -1 1 1
        1 1 -1 -1 -1 -1 -1 -1 1 1 -1 -1 -1 1 -1 -1 1 -1 -1 1 -1 -1 -1 -1 1 1 1 1 -1 1 -1
        -1 1 1 -1 1 -1 1 1 -1 -1 -1 -1 1 -1 -1 1 1 1 -1 1 -1 -1 -1 -1 -1 1 1 1 -1 -1 -1
        1 -1 -1 -1 -1 -1 1 -1 -1 -1 -1 -1 1 -1 1 1 1 -1 -1 1 -1 1 1 1 1 -1 -1 1 -1 -1 1
        1 -1 1 -1 1 -1 1 -1 -1 1 -1 -1 1 -1 1 1 1 1 1 -1 -1 -1 1 1 -1 -1 1 1 1 1 -1
        1 1 1 -1 -1 -1 1 -1 1 -1 -1 -1 -1 1 -1 -1 -1 1 1 -1 1 1 -1 -1 -1 1 -1 -1 -1 -1 1
        -1 -1 -1 -1 1 -1 1 -1 1 -1 1 1 1 -1 -1 1 -1 -1 1 1 1 1 1 1 -1 1 1 1 -1 -1 -1
        1 1 -1 1 -1 -1 -1 -1 1 1 1 -1 1 -1 -1 1 1 -1 1 -1 1 1 1 1 1 1 -1 1 -1 1 -1
        1 -1 -1 -1 -1 1 1 -1 1 1 -1 1 1 1 -1 -1 1 -1 1 -1 -1 1 1 1 -1 1 1 1 -1 1 1
        -1 -1 -1 -1 1 -1 -1 1 1 -1 1 -1 1 1 -1 1 1 -1 1 1 1 -1 -1 -1 -1 1 -1 -1 -1 -1 -1
        -1 1 -1 -1 -1 1 -1 1 1 -1 1 -1 -1 -1 -1 1 1 1 -1 -1 -1 1 -1 -1 -1 1 -1 1 -1 -1 -1
        1 1 1 1 1 -1 -1 1 1 1 1 -1 1 1 1 -1 -1 -1 -1 -1 1 -1 1 1 -1 -1 1 1 -1 -1 1];
    code(find(code==-1))=0;
elseif codeselector==12
    codetype='Gold';
    mseq1=mSeqGen([1 1 1 0 1 1 1 1 1],oct2bin([1 0 2 1]));
    mseq2=mSeqGen([1 1 0 1 1 1 1 0 1],oct2bin([1 1 3 1]));
    code=gold_gen(mseq1,mseq2);
elseif codeselector==13  %MPS
    codetype='MPS';
    code=[0  0 0 0 0  0 1 0 0  0 1 0 1  1 0 1 0  0 1 1 0  1 0 1 1 ...
             0 0 0 1  1 1 0 1  0 0 1 1  1 0 1 1  0 0 0 1  1 0 1 0 ...
             1 1 1 0  0 0 0 0  1 0 1 1  1 0 1 1  1 0 1 1];
    %code(find(code==-1))=0;
```

```
elseif codeselector==14  %SIMULATED ANNEALING
    %SA_69.txt
    %Thursday, 12/5/2002, 18:12
    %N: 69, M: 15,  Number of Phases: 2
    %Code Type: Binary
    %Initial Temperature: 310.00 , Final Temperature: 0.03
    %Temp decrement: T' = 0.10T
    %Equilibrium: #accept or #reject > NM(1+0.1NM/T)
    %Cost Function: Max Sidelobes; Auto Weight: 1.00, Cross Weight: 1.00
    %Emin: 33.000000
    codetype='SAAC';
    code=[-1 1 -1 -1 -1 -1 -1 1 1 -1 -1 -1 -1 1 1 -1 1 -1 1 -1 1 1 1 1 -1 1 -1 1 -1 -1 1 1 1 -1 ...
            -1 -1 -1 1 -1 -1 -1 1 1 -1 1 1 -1 1 1 -1 1 1 1 -1 -1 -1 1 1 1 -1 1 -1 -1 -1 1 1 1 1 1
        1 1 -1 1 -1 -1 1 -1 -1 1 1 1 -1 -1 1 -1 -1 1 -1 -1 -1 -1 1 1 1 1 -1 1 -1 1 1 1 -1 -1 -1 ...
            -1 1 -1 1 1 1 -1 -1 -1 -1 -1 -1 -1 1 1 -1 1 1 -1 1 1 -1 1 1 1 1 -1 1 1 1 -1 1 1 1
        1 1 1 -1 1 -1 -1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 1 1 1 -1 1 -1 -1 -1 -1 -1 -1 1 -1 -1 1 1 1 ...
            1 1 1 -1 1 -1 -1 -1 1 -1 -1 -1 1 1 1 -1 -1 1 1 -1 -1 1 1 -1 1 1 -1 1 1 -1 1 -1 1 -1
        1 -1 -1 1 -1 -1 -1 1 1 1 1 1 1 1 1 1 -1 1 1 -1 1 -1 -1 -1 1 -1 1 -1 1 1 1 -1 1 1 1 1 1 1 ...
            1 -1 -1 1 1 -1 1 1 1 -1 -1 -1 -1 -1 1 -1 1 1 -1 1 -1 1 -1 -1 -1 -1 -1 1 -1 -1 1
        1 -1 1 -1 -1 -1 1 1 1 -1 -1 -1 1 -1 1 1 1 -1 -1 1 -1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 -1 ...
            -1 1 -1 -1 -1 -1 1 1 1 -1 1 -1 -1 1 1 -1 1 1 1 1 1 1 -1 -1 -1 1 -1 -1 1 -1 -1 -1 -1 -1 1
        -1 -1 -1 1 -1 1 -1 1 1 1 1 1 1 -1 1 -1 1 -1 1 1 -1 -1 -1 1 1 -1 -1 1 1 -1 -1 -1 1 -1 1 -1 -1 ...
            -1 1 1 -1 1 -1 -1 1 -1 1 -1 -1 1 1 -1 1 -1 1 1 1 1 1 1 -1 -1 -1 -1 -1 1 -1 1 1
        -1 1 1 -1 -1 -1 1 1 -1 -1 -1 -1 -1 -1 1 -1 1 -1 -1 1 -1 -1 1 1 -1 1 1 1 1 -1 1 -1 -1 -1 -1 -1 ...
            1 -1 -1 -1 -1 1 1 1 1 -1 1 1 1 -1 -1 -1 1 1 -1 1 -1 -1 -1 1 -1 -1 1 -1 -1 1 1 1 -1
        1 1 -1 -1 1 -1 1 -1 1 1 1 1 1 1 -1 -1 1 1 -1 -1 1 -1 1 1 1 -1 1 -1 -1 -1 -1 1 1 -1 -1 -1 -1 ...
            -1 -1 -1 1 1 1 1 1 1 -1 -1 1 1 1 -1 -1 1 -1 -1 1 1 1 -1 1 -1 1 -1 1 1 1 -1 -1
        1 -1 1 -1 1 1 1 -1 -1 1 -1 1 1 -1 -1 -1 -1 1 -1 1 1 1 -1 -1 -1 1 -1 1 -1 1 1 1 -1 1 -1 -1 -1 ...
            -1 1 -1 1 1 -1 -1 1 -1 -1 1 1 -1 1 1 -1 1 -1 -1 1 1 1 1 -1 1 1 1 1 -1 -1 -1 -1
        -1 -1 -1 -1 1 -1 -1 1 -1 -1 -1 -1 1 1 -1 1 1 -1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 -1 1 1 -1 1 ...
            1 1 1 1 -1 1 -1 1 -1 1 -1 -1 1 1 -1 -1 -1 1 1 -1 1 -1 -1 -1 1 -1 1 -1 -1 -1 -1 1 1
        -1 1 -1 -1 1 1 -1 -1 1 -1 1 1 1 1 1 1 1 -1 1 -1 1 1 -1 1 1 1 1 -1 -1 -1 -1 1 -1 1 1 1 -1 1 -1 ...
            -1 1 -1 1 1 1 -1 -1 1 1 -1 -1 -1 1 -1 1 1 -1 1 1 1 1 1 -1 1 -1 -1 -1 -1 -1
        1 1 1 -1 1 1 1 1 -1 1 -1 -1 -1 -1 1 1 1 -1 1 1 -1 1 1 -1 1 1 -1 1 1 -1 -1 1 1 -1 -1 1 -1 -1 -1 ...
            -1 -1 1 1 -1 1 1 1 -1 -1 -1 1 -1 1 1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 -1 1
        -1 1 1 1 1 -1 -1 1 -1 -1 1 1 -1 -1 -1 1 -1 -1 -1 1 1 1 -1 -1 1 -1 1 -1 1 1 -1 -1 -1 -1 1 -1 1 ...
            -1 -1 -1 -1 1 -1 -1 -1 -1 1 -1 -1 -1 1 -1 1 -1 -1 -1 -1 -1 1 1 -1 -1 1 1 -1 1 1 -1 1
        -1 -1 -1 -1 -1 -1 1 -1 -1 1 1 -1 -1 1 1 1 1 -1 1 -1 1 -1 1 1 1 1 -1 -1 -1 -1 -1 1 -1 -1 1 -1 ...
            1 1 -1 -1 -1 -1 1 1 -1 1 1 -1 1 1 -1 1 -1 -1 -1 -1 1 -1 1 1 1 1 1 -1 1 1 -1 -1 1
        -1 1 -1 -1 1 -1 1 1 -1 -1 -1 1 1 -1 1 1 -1 1 -1 1 1 -1 1 1 -1 -1 -1 1 -1 1 -1 -1 -1 -1 -1 -1 ...
            -1 -1 -1 1 1 -1 -1 -1 1 -1 1 -1 -1 1 1 -1 1 -1 1 -1 1 -1 1 -1 -1 1 1 1 1 1 1 1 -1];
    code(find(code==-1))=0;

elseif codeselector==15  %SIMULATED ANNEALING
    %SAC code modified to have first 5 rows on bottom of code
    codetype='SACmod';
    code=[1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 -1 -1 1 -1 1 -1 1 -1 1 1 1 -1 -1 1 1 -1 1 -1
        -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 -1 1 -1 1 1 1 1 1 1 1 1 1 -1 -1 1 1 -1 1 -1
        -1 -1 -1 -1 -1 1 -1 -1 -1 -1 -1 1 1 1 1 -1 1 1 -1 -1 1 1 1 -1 -1 -1 -1 -1 -1 1 -1
        -1 1 -1 -1 -1 -1 1 -1 -1 1 -1 -1 -1 1 1 -1 1 1 1 1 1 -1 -1 -1 -1 1 1 -1 -1 1 -1
        1 1 1 -1 1 -1 1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 1 -1 1 -1 1 1 -1 1 1 -1 -1 -1
        -1 -1 -1 -1 1 -1 1 -1 1 -1 1 1 -1 -1 1 -1 -1 -1 -1 1 1 1 -1 1 -1 -1 -1 1 -1 -1 -1
        -1 1 1 1 1 -1 -1 1 1 1 -1 -1 -1 1 1 -1 -1 -1 -1 1 -1 -1 1 -1 1 1 -1 1 1 1 -1
        -1 -1 -1 1 -1 1 1 -1 -1 1 1 1 1 -1 1 1 1 -1 -1 1 1 1 -1 1 -1 1 -1 -1 1 1 -1
        1 1 -1 -1 -1 1 1 -1 -1 -1 1 -1 -1 1 1 1 -1 1 -1 -1 1 -1 -1 -1 1 1 -1 1 1 1 1
        1 -1 1 1 1 1 1 1 -1 -1 1 1 1 1 1 -1 -1 -1 1 -1 -1 1 1 1 -1 -1 -1 -1 -1 -1 -1
        -1 -1 -1 -1 1 1 1 -1 1 -1 1 1 1 -1 -1 -1 1 1 1 1 1 -1 -1 1 1 1 1 1 1 -1 1
        -1 -1 -1 1 1 1 1 -1 1 1 1 -1 -1 -1 -1 -1 -1 1 1 1 -1 1 1 1 1 -1 1 1 1 1 -1
        -1 1 1 -1 1 1 -1 -1 -1 1 -1 -1 1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 1 -1 1 1 -1
        -1 -1 -1 1 1 -1 -1 1 1 -1 1 1 -1 1 1 1 -1 1 1 -1 -1 1 1 1 1 -1 -1 1 -1 -1 -1
        -1 -1 -1 1 -1 1 -1 1 -1 -1 -1 -1 1 1 -1 -1 1 -1 1 -1 -1 -1 -1 -1 -1 1 1 -1 1 -1 -1 ];
    code(find(code==-1))=0;

elseif codeselector==50  %Temporary Testing
    %1)Use SAC code1&code2 and run through gold_gen to find code metrics
    codetype='TempTest';
    code=[-1 -1 -1 -1 1 1 1 -1 1 -1 1 1 1 -1 -1 -1 1 1 1 1 1 -1 -1 1 1 1 1 1 1 -1 1
        -1 -1 -1 1 1 1 1 -1 1 1 1 -1 -1 -1 -1 -1 -1 1 1 1 -1 1 1 1 1 -1 1 1 1 1 -1
        -1 1 1 -1 1 1 -1 -1 -1 1 -1 -1 1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 1 -1 1 1 -1
        -1 -1 -1 1 1 -1 -1 1 1 -1 1 1 -1 1 1 1 -1 1 1 -1 -1 1 1 1 1 -1 -1 1 -1 -1 -1
        -1 -1 -1 1 -1 1 -1 1 -1 -1 -1 -1 1 1 -1 -1 1 -1 1 -1 -1 -1 -1 -1 -1 1 1 -1 1 -1 -1
        1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 -1 -1 1 -1 1 -1 1 -1 1 1 1 -1 -1 1 1 -1 1 -1
        -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 -1 1 -1 1 1 1 1 1 1 1 1 1 -1 -1 1 1 -1 1 -1
        -1 -1 -1 -1 -1 1 -1 -1 -1 -1 -1 1 1 1 1 -1 1 1 -1 -1 1 1 1 -1 -1 -1 -1 -1 -1 1 -1
        -1 1 -1 -1 -1 -1 1 -1 -1 1 -1 -1 -1 1 1 -1 1 1 1 1 1 -1 -1 -1 -1 1 1 -1 -1 1 -1
        1 1 1 -1 1 -1 1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 1 -1 1 -1 1 1 -1 1 1 -1 -1 -1
        -1 -1 -1 -1 1 -1 1 -1 1 -1 1 1 -1 -1 1 -1 -1 -1 -1 1 1 1 -1 1 -1 -1 -1 1 -1 -1 -1
        -1 1 1 1 1 -1 -1 1 1 1 -1 -1 -1 1 1 -1 -1 -1 -1 1 -1 -1 1 -1 1 1 -1 1 1 1 -1
        -1 -1 -1 1 -1 1 1 -1 -1 1 1 1 1 -1 1 1 1 -1 -1 1 1 1 -1 1 -1 1 -1 -1 1 1 -1
        1 1 -1 -1 -1 1 1 -1 -1 -1 1 -1 -1 1 1 1 -1 1 -1 -1 1 -1 -1 -1 1 1 -1 1 1 1 1
        1 -1 1 1 1 1 1 1 -1 -1 1 1 1 1 1 -1 -1 -1 1 -1 -1 1 1 1 -1 -1 -1 -1 -1 -1 -1   ];
    code=gold_gen(code(1,:),code(2,:));
    code(find(code==-1))=0;

end lenc=length(code(1,:));

% for i=1:lenc %Determine decimal values of binary code:
%     fprintf('%4e\n',bin2dec(sprintf('%d',code(i,:))))
% end
```

B.3 plotdata.m

```
function
plotdata(Vt,row,dt,Twp,T,To,Tc,ref_matrix,NBI,Rcj,ds_mod_tx,th_mod_tx)
%
%plotdata(nb,showmatrix,row,lenVt,Tc,pultr,To,Vt,...
%           ref_pos,ref_neg,ref_ppm0,ref_ppm1,ds_mod_tx,th_mod_tx)
%
%This function created by Courtney M. Canadeo
%on 13 Sept. 2002
%
pultr=repmat(Vt(1:Tc),1,row); %Pulse train
u=0:dt:(To-dt);               %Time for single pulse
lenVt=length(Vt);             %Length of Gaussian pulse
alpha=50;

close all;
%%%PLOT 1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(5); plot(pultr); hold off; title(’Received Waveform (In
Correlator)’); disp(’Plotting Pulses... ’);

%%%PLOT 2%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%create plot of PSD:
figure(7); x=Vt(1:Tc); t=0:dt:Tc*dt-dt;
%fftlen=2^nextpow2(d);
%fftfreqax=[-1/(2*dt):(1/dt)/fftlen:1/(2*dt)-1];
%semilogy(fftfreqax,fftshift(abs((1/sqrt(length(Vt(1:d)))*fft(Vt(1:d),fftlen)).^2)));
subplot(2,1,1) plot(t,x);title(’Transmitted Waveform (In
Channel)’);

subplot(2,1,2)
Fx=2/length(x)*fftshift(abs(fft(x,2^(4+nextpow2(length(x))))));
F=1/(2*dt); df=F*2/(length(Fx)); f_index=[-F+df/2:df:F-df/2];
%f_index=[-F+df:df:F];
plot(f_index,Fx);title(’Transmitted Frequency (In Channel)’);

%figure(8)
%lim=20e9;f=-lim:lim/10:lim;
%Wf=-j/2*1e18*T^2*exp(-.5*(f*T).^2); %See: Maggio, PCTH
%semilogy(f,Wf.^2);
disp(’Plotting PSD...’);


%%%PLOT 3%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(11); subplot(2,2,1);plot(ref_matrix(1,:));title(’refBPSK0’)
subplot(2,2,2);plot(ref_matrix(2,:));title(’refBPSK1’)
subplot(2,2,3);plot(ref_matrix(3,:));title(’refPPM0’)
subplot(2,2,4);plot(ref_matrix(4,:));title(’refPPM1’)
disp(’Plotting Pulse References...’);

%%%PLOT 4%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nargin>=9
%create plot of NBI:
figure(13); Tcj=round(1/(Rcj*dt));
NBI=NBI’;x=NBI(1:Tcj);
t=0:dt:dt*length(x)-dt; subplot(2,1,1) plot(t,x);title(’NBI
Waveform (In Channel)’);

subplot(2,1,2)
Fx=2/length(x)*fftshift(abs(fft(x,2^(3+nextpow2(length(x))))));
F=1/(2*dt); df=F*2/(length(Fx)); f_index=[-F+df/2:df:F-df/2];
plot(f_index,Fx,’--’);title(’NBI Frequency (In Channel)’);
disp(’Plotting NBI...’);

figure(7);hold on; subplot(2,1,2) plot(f_index,Fx,’--’); end
figure(7)%Put focus on freq. plot
```

B.4 cmc_codestat.m

```
%function cmc_codestat()
%cmc_codestat.m
%20 Nov. 2002
%
%This code primarily for DS-BPSK case.
%
%Periodic correlation metrics for binary codes:
%PSL:   Peak Side-lobe Level
```

```
%ISL:   Integrated Side-lobe Level
%PCCL:  Peak Cross Correlation Level
%
%Include:   codeselect.m
%           codestat.m....by Capt. Kevin Sitler
%           Disc_XCorr.m
clear all;
for nb_users=2:15;  %Changed from=15 to =2:15 for peaks/user level
    diary(['data/codemetricsUsers_1.txt']);
    disp(' ');
    disp(['NbUsers= ' num2str(nb_users) ]);%'  Code length = ' num2str(lenc) ' bits.']);
    diary off;

    for codeselector=[2 3 4 5 8 15]
        [code codetype lenc]=codeselect(codeselector);
        %%%Code -> Unipolar

        %Initialize:
        cc(nb_users,nb_users,lenc)=0;

        diary(['data/codemetricsUsers_1' num2str(codetype) num2str(lenc) '.txt']);
        disp('');
        disp(['NbUsers= ' num2str(nb_users) '  Code length = ' num2str(lenc) ' bits.']);

        %Uncomment %nb_users for m=1:1 below to run for all users in code family
        for m=1:1 %:nb_users %:1
            for n=m:nb_users

                cc(m,n,:)=Disc_XCorr(code(m,:),code(n,:));
                %This creates normalized periodic correlation.

                if m==n
                    disp(['m= ' num2str(m)]);
                    %Used for periodic correlations:
                    ac(m,:)=Disc_XCorr(code(m,:),code(n,:));
                    %  Use next for aperiodic correlations:
                    %  ac=conv(code,fliplr(code));
                    %Remove peak correlation,determine side-lobe values:
                    ac(:,lenc)= 0;
                    pslp(m) = max(ac(m,:));
                    psln(m) = min(ac(m,:));
                    pslm(m) = max(abs(ac(m,:)));
                    isl(m)=sum(ac(m,:).^2);
                    isldb(m)=10*log10(isl(m));

                    disp(['Code ' num2str(m) ':']);
                    %disp(['  PSL Amplitude:  max = ' num2str(pslp(m)) ' (' ...
                    %                                num2str(20*log10(pslp(m))) 'dB)']);
                    %disp(['                  min = ' num2str(psln(m)) ]);
                    disp(['  PSL Magnitude:  max = ' num2str(pslm(m)) ' (' ...
                            num2str(20*log10(pslm(m))) 'dB)']);
                    disp(['  ISL:                  ' num2str(isldb(m)) 'dB']);

                else

                    pcclp(m,n) = max(cc(m,n,:));
                    pccln(m,n) = min(cc(m,n,:));
                    pcclm(m,n) = max(abs(cc(m,n,:)));

                    disp(['    Codes ' num2str(m) ' & ' num2str(n) ':']);
                    %disp(['    PCCL  Amplitude: max = ' num2str(pcclp(m,n)) ' (' ...
                    %       num2str(20*log10(pcclp(m,n))) 'dB)']);
                    %disp(['                     min = ' num2str(pccln(m,n)) ]);
                    disp(['    PCCL  Magnitude: max = ' num2str(pcclm(m,n)) ' (' ...
                            num2str(20*log10(pcclm(m,n))) 'dB)']);
                end
            end
        end
        diary off;

%         figure(900+codeselector)
%         subplot(3,1,1)
%         hist(pslm(find(pslm>0)),[0:1/lenc:1]);hold on;
%         title(['DS-AutoCorr' num2str(codetype)]);ylabel('Max PSL');
%         subplot(3,1,2)
%         hist(isl(find(isl>0)),[0:1/lenc^3:1/lenc^2]);hold on;
%         ylabel('ISL');
%         subplot(3,1,3)
%         hist(pcclm(find(pcclm>0)),[0:1/lenc:1]);hold on;
%         xlabel('Bin');ylabel('PCCL');

        diary('data/codemetricsUsers_1.txt');
        disp([num2str(codetype) num2str(lenc) '...PSL  Magnitude: MAX: ' ...
                num2str(max(20*log10(pslm))) '   MIN: ' num2str(min(20*log10(pslm)))]);
        disp([num2str(codetype) num2str(lenc) '...ISL  Magnitude: MAX: ' ...
                num2str(max(isldb)) '   MIN: ' num2str(min(isldb))]);
        disp([num2str(codetype) num2str(lenc) '...PCCL Magnitude: MAX: ' ...
                num2str(20*log10(max(max(pcclm(find(pcclm~=0)))))) '   MIN: ' ...
                num2str(20*log10(min(min(pcclm(find(pcclm~=0))))))]);
        diary off;
    end
end
%code(find(code==0))=-1;
%%%Code -> Bipolar
```

B.5 agreeSeq.m

```
%agreeSeq.m
%Courtney Canadeo
%21 Nov. 2002
%
%Determines the number of terms that agree and disagree between
%each member of a code family. This function is primarily designed
%for case when elements of code are mapped to non-binary numbers.
clear all; Tc=50;
for nb_users=1:15
 for rolap=1
    diary('data/agreeSeqUSERS_1.txt');
    disp(' ');
    disp(['nbusers: ' num2str(nb_users)]);
    diary off;

    for codeselector=[2 8 3 4 5 15]
        [code codetype lenc]=codeselect(codeselector);
        %%%Code <-> Unipolar

        fa(1:lenc)=0;
        th_ck=zeros(nb_users,lenc);
        for j=1:nb_users
            for i=1:lenc
                regsz=5;
                %r=mod(i-1,lenc);
                r=mod(rolap*i-rolap,lenc);
                if r<=(lenc-regsz)
                    th_ck(j,i)=bin2dec(sprintf('%d',code(j,r+1:r+regsz))) *Tc;
                else
                    th_ck(j,i)=bin2dec(sprintf('%d%d',code(j,r+1:lenc),code(j,1:r+regsz-lenc))) *Tc;
                end
            end
        end

        %Initialize:
        %diary(['data/agreeSeq_15' num2str(rolap) num2str(codetype) num2str(lenc) '.txt']);
        disp(['rolap: ' num2str(rolap) '  Code length = ' num2str(lenc) ' bits.']);
%THIS SECTION OF CODE USER FOR DETERMINING NUMBER OF AGREE PER TIME SLOT
%Uncomment (:nb_users-1) to determine #collisions occurring in network,
%not just with User1 (signal of interest).
        for slot=1:lenc
            for codenb=1:1   %(nb_users-1)
                na_temp(codenb)=sum(th_ck(codenb,slot)==th_ck(codenb+1:nb_users,slot));
                nd_temp(codenb)=sum(th_ck(codenb,slot)~=th_ck(codenb+1:nb_users,slot));
            end
            na(slot)=sum(na_temp);
            nd(slot)=sum(nd_temp);
        end
        diary off;

        diary('data/agreeSeqUSERS_1.txt');
        %disp([num2str(codetype) num2str(lenc) sprintf('\t') 'Na:  ' num2str(sum(sum(na))) ...
        %                            sprintf('\t') 'Nd: ' num2str(sum(sum(nd)))]);
        disp([num2str(codetype) num2str(lenc) sprintf('\t') 'Na:  ' num2str(sum(na)) ...
                sprintf('\t') 'Nd: ' num2str(sum(sum(nd)))]);
        diary off;

        figure(1100+codeselector);
        plot(1:lenc,na,'s-');
        %plot(1:lenc,fa,'s-');
        title(['TH-NbAgree: ' num2str(codetype)]);xlabel('Code Index');ylabel('N_A');
    end
end
end%nb_users loop
```

Bibliography

[1] Aiello, G. Robert, et al. *Ultra-wideband: An emerging technology for wireless communications*. Conference Proceedings of the IEEE VTC-2001, Rhodes, Greece: Fantasma Networks, Inc., May 2001.

[2] Anderson, Jon M. *Nonlinear Supression of Range Ambiguity in Pulse Doppler Radar*. PhD Dissertation, AFIT/DS/ENG/01-05, Graduate School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 2001.

[3] Barrett, Terence W. History of UltraWideBand (UWB) Radar & Communications: Pioneers and Innovators. Progress In Electromagnetics Symposium 2000 (PIERS2000). Cambridge,MA, July 2000.

[4] Canadeo, Courtney M., et al. "Code Selection For Enhancing UWB Multiple Access Communication Performance Using TH-PPM and DS-BPSK Modulations," *IEEE Wireless Communications Networking Conference* (March 2003).

[5] Cassioli, Dajana, et al. *A Statistical Model for the UWB Indoor Channel*. Conference Proceedings of the IEEE VTC-2001, May 2001.

[6] Dillard, Wade E. "Wireless Area Detection Enclosures for Geo-location of Wireless Assets." Address to AFIT students. Air Force Institute of Technology (AETC), Wright-Patterson AFB OH, 12 December 2002.

[7] Durisi, Giuseppe. *On the Validity of Gaussian Approximation to Characterize the Multiuser Capacity of UWB TH PPM*. IEEE Conference on UWB Systems and Technologies, Politecnico di Torino; Giovanni Romano, Telecom Italia Lab S.p.A., ITALY, May 2002.

[8] Federal Communications Commission. *First Report and Order: Revision of Part 15 of the Commission's Rules Regarding Ultra-Wideband Transmission Systems*. Report Series ET Docket 98-153. Washington: Government Printing Office, April 2002.

[9] Foerster, Jeff, et al. *Impact of Ultra Wide Band Transmissions on a Generic Receiver*. Journal Q2, Intel Architecture Labs, 2001.

[10] Foerster, Jeffrey R. *The Effects of Multipath Interference on the Performance of UWB Systems in an Indoor Wireless Channel*. Conference Proceedings of the IEEE VTC-2001, May 2001.

[11] Foerster, Jeffrey R. *The Performance of a Direct-Sequence Spread Spectrum Ultra-WideBand System in the Presence of Multipath, Narrowband, Interference, and Multiuser Interference*. IEEE Conference on Ultra Wideband Systems and Technologies, May 2002.

[12] Fontana, Robert J. *A Note on Power Spectral Density Calculations for Jittered Pulse Trains*. Multispectral Solutions, INC., 2000.

[13] Fontana, Robert J. *On Range-Bandwidth per Joule for Ultra Wideband and Spread Spectrum Waveforms*. Multispectral Solutions, INC., July 2000.

[14] Fontana, Robert J. and Steven J. Gunderson. *Ultra-Wideband Precision Asset Location System*. IEEE Conference on UWB Systems and Technologies, Multispectral Solutions, INC., and Naval Facilities Engineering Service Center, May 2002.

[15] Gardner, William A. "Signal Interception: A Unifying Theoretical Framework for Feature Detection," *IEEE Transactions on Communications*, *36*(8) (August 1988).

[16] Gardner, William A. "Exploitation of Spectral Redundancy in Cyclostationary Signals," *IEEE SP Magazine* (April 1991).

[17] Hoctor, Ralph and Harold Tomlinson. *Delay-Hopped Transmitted-Reference RF Communications*. GE Global Research, 2002.

[18] Jain, Raj. *The Art of Computer Systems Performance Analysis*. New York: John Wiley & Sons, Inc., 1991.

[19] LAN/MAN Standards Committee of the IEEE Computer Society. *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. ANSI/IEEE Std 802.11 ISO/IEC 8802-11:1999(E). 1999.

[20] Maggio, Gian Mario, et al. "Pseudo-Chaotic Time Hopping For UWB Impulse Radio," *28*(12) (December 2001).

[21] Nathanson, F. E. *Radar Design Principles*. Mendham,New Jersey: SciTech, 1999.

[22] National Telecommunications and Information Administration. *The Temporal and Spectral Characteristics of Ultrawideband Signals*. Report Series 01-383. Washington: Government Printing Office, January 2001.

[23] Peterson, Roger, et al. *Introduction to Spread Spectrum Communications*. Prentice Hall, 1995.

[24] Ramirez-Mireles, Fernando. "Performance of Ultrawideband SSMA Using Time Hopping and M-ary PPM," *IEEE Journal on Selected Areas in Communications*, *19*(6) (June 2001).

[25] Sarwate, D.V. and M. B. Pursley. "Crosscorrelation Properties of Pseudorandom and Related Sequences," *Proceedings of the IEEE*, *68*:593–619 (May 1980).

[26] Scholtz, R. A. *Multiple Access with Time-Hopping Impulse Modulation*. IEEE MILCOM'93, October 1993.

[27] Shannon, C. E. *A mathematical theory of communication*. Bell System Technical Journal, October 1948.

[28] Shapiro, S. S. and M. B. Wilk. "An Analysis of Variance Test for Normality (Complete Samples)," *Biometrika*, (52,53,54):591–611 (1965).

[29] Siwiak, Kazimierz. *Impact of Ultra Wide Band Transmissions on a Generic Receiver*. Conference Proceedings of the IEEE VTC-2001, Rhodes Greece: Time Domain Corporation, May 2001.

[30] Siwiak, Kazimierz, et al. *Ultra-Wide Band Radio: The Emergence of an Important New Technology*. Conference Proceedings of the IEEE VTC-2001, Huntsville AL: Time Domain Corporation, 2001.

[31] Sklar, Bernard. *Digital Communications*. New Jersey: Prentice Hall, 2001.

[32] Somayazulu, V. Srinivasa. *Multiple Access Performance in UWB Systems using Time Hopping vs. Direct Sequence Spreading*. Intel Labs, 2002.

[33] Telatar, I. Emre and David N. C. Tse. "Capacity and Mutual Information of Wideband Multipath Fading Channels," *46*(4) (July 2000).

[34] Zeisberg, S., et al. *Performance Limits of Ultra-Wideband Basic Modulation Principles*. Communications Laboratory, Dresden University of Technology, Germany, 2001.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)*
March 2003

2. REPORT TYPE
Master's Thesis

3. DATES COVERED *(From - To)*

4. TITLE AND SUBTITLE
ULTRA WIDE BAND MULTIPLE ACCESS PERFORMANCE USING TH-PPM AND DS-BPSK MODULATIONS

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Canadeo, Courtney M., Second Lieutenant, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, Building 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GE/ENG/03-06

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Attn: James P. Stephens
AFRL/SNRW
WPAFB OH 45433-7765
DSN: 785-5579 x4239
james.stephens@wpafb.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES
AFIT Technical POC: Michael A. Temple, AFIT/ENG
michael.temple@afit.edu

14. ABSTRACT
The increasing demand for portable, high data rate communications has focused attention on wireless technology. Ultra Wide Band (UWB) waveforms have the ability to deliver megabits of information while maintaining low average power consumption. In accordance with recent FCC rulings, UWB systems are now allowed to operate in the unlicensed spectrum of 3.1 to 10.6 GHz, motivating renewed interest in the forty year old concept of impulse radio. Gaussian monocycles produce UWB waveforms occupying large bandwidths with multiple access (MA) capability enabled by spread spectrum techniques. Time Hopping (TH) and Direct Sequence (DS) modulations are considered here for UWB MA applications. This work extends Gold coding results and characterizes UWB performance using Simulated Annealing (SA) and Random Integer (RI) codes for TH and DS UWB applications. TH-PPM and DS-BPSK performance is evaluated using simulated probability of bit error Pb under MA interference (MAI), multipath interference (MPI), and narrow band interference (NBI) conditions for synchronous and asynchronous networks. Communication performance is validated for a single user operating over an AWGN channel and extended to incorporate MA capability.

15. SUBJECT TERMS
Ultra Wide Band (UWB), communication performance, Multiple Access Interference (MAI), Time Hopping Pulse Position Modulation (TH-PPM), Direct Sequence Binary Phase Shift Keying (DS-BPSK), synchronous networks, Gold code.

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Michael A. Temple, AFIT/ENG
U	U	U	UU	124	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-3636, ext 4703